W LC ME TO AUST N

CONNECT

Building a New Design Methodology for a Large Channel Count Application

Overview and Lessons Learned

Jeff Tipps

Principal Software Engineer

1. Application Space
2. Problem to Solve
3. Using MBSE to Gather Industry Requirements
4. Hardware Architecture
5. Software Suite
6. Validating the Architecture
7. Project Deliverables
8. Lessons Learned

Application Space

Firefly Alpha Rocket
100 channels strain, displacement, temperature

Boeing Wing Flex Test
100’s of strain channels

NASA/Boeing SLS
2,000 channels strain, displacement, pressure

Static Structural Test Considerations

Many strain channels

- Thousands of channels to get all data

Data Synchronization

- Can data points across entire DUT

Redundancy

- Resiliency against sensor and equipment failure

Distributed across long distances

- Avoid ground loops, signal noise

Team Monitoring

- Tests monitored by team of subject matter experts

Pre-Test System Checks

- Validate test equipment before running the test

Problem(s) to Solve:

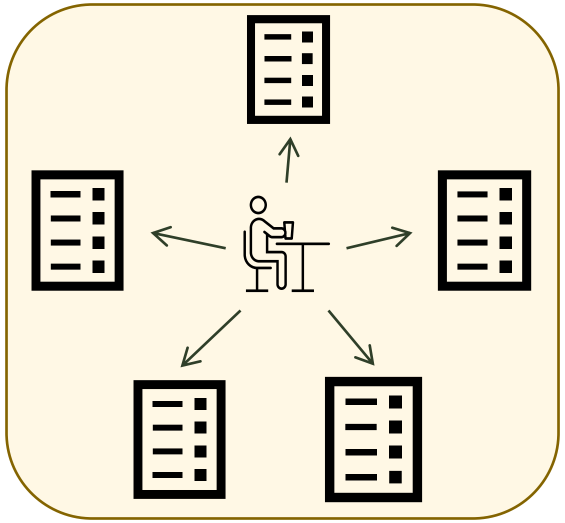
- NI’s hardware platforms have similar (but not identical) channel type support
  - PXI
  - C Series / FieldDAQ
- Assume you pick one (cDAQ). How do you synchronize?
  - 9469
  - TSN
- Assume you pick one (TSN). What tool do you choose?
  - [illegible]ine
  - [illegible]ing
  - [illegible]ar
- Assume you pick one. What software do you use?
  - LabVIEW (write your own)
  - FlexLogger
  - Partner Solutions
- Assume you go through all of this. How well will it work?

CONFIDENCE

Must Identify Common Industry Requirements

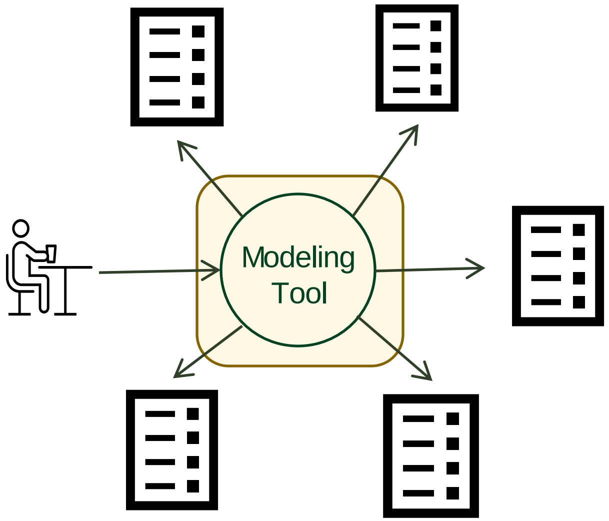
We did a little Model Based Systems Engineering. What is MBSE?

Document-Based



System Definition is the sum of the documents

Model-Based

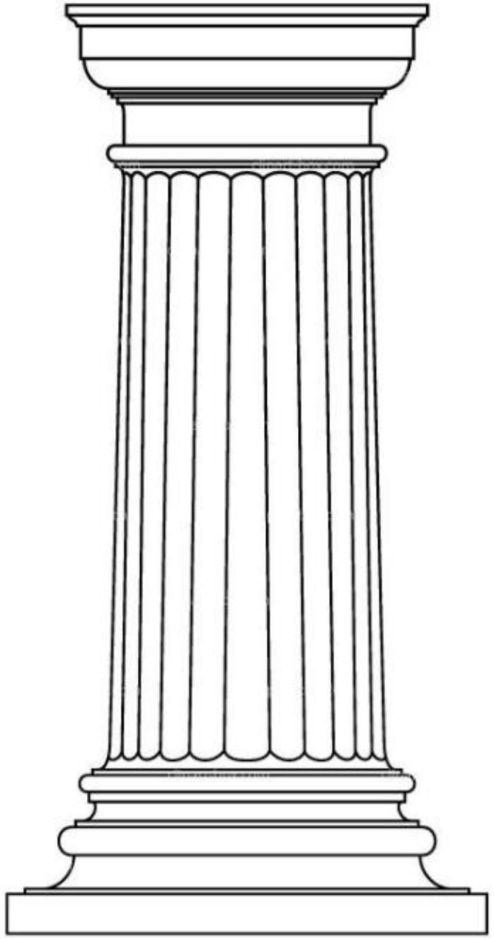


System Definition is captured by a model in a modeling tool

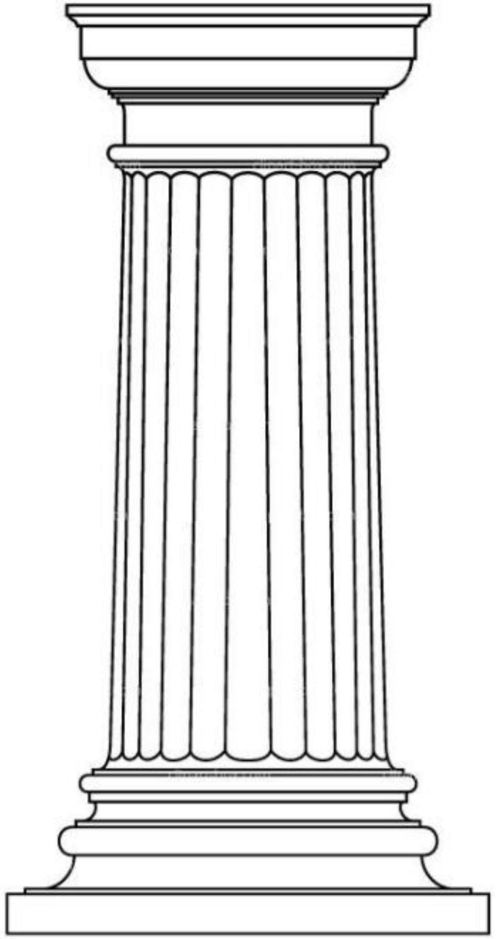
Documents are *scripted*

Three Pillars of MBSE

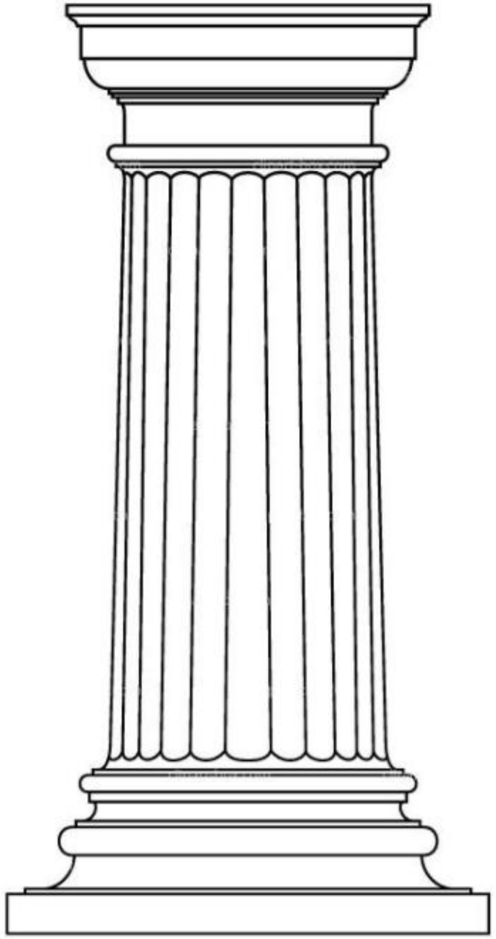
LANGUAGE



TOOL



METHODOLOGY



Three Pillars of MBSE

LANGUAGE

SYSML

OMG SYSTEMS MODELING LANGUAGE SYSML™

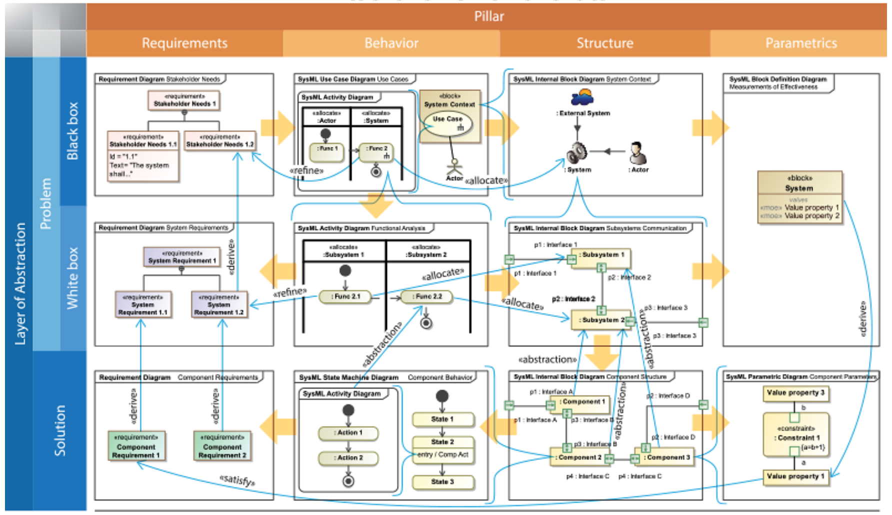
TOOL

Cameo

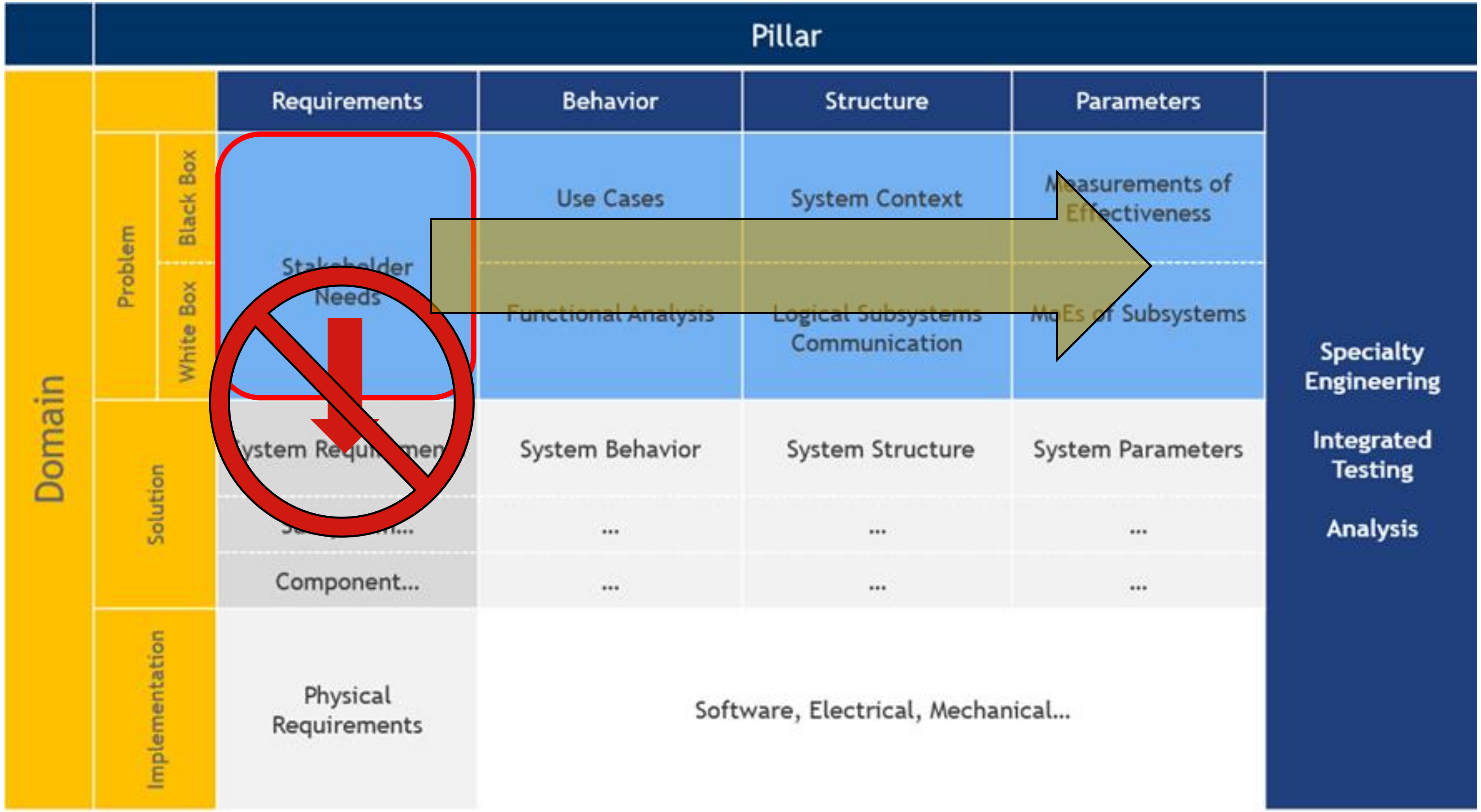
CAMEO SYSTEMS MODELER™

METHODOLOGY

MBSE Grid


Pillar
Requirements
Behavior
Structure
Parametrics
Layer of Abstraction
Problem
Black box
White box
Solution
Requirement Diagram Stakeholder Needs
«requirement» Stakeholder Needs 1
«requirement» Stakeholder Needs 1.1
Id = "1.1"
Text= "The system shall..."
«requirement» Stakeholder Needs 1.2
SysML Use Case Diagram Use Cases
SysML Activity Diagram
«allocate» :Actor
«allocate» :System
: Func 1
: Func 2
«block» System Context
Use Case
Actor
«refine»
«allocate»
SysML Internal Block Diagram System Context
: External System
: System
: Actor
SysML Block Definition Diagram Measurements of Effectiveness
«block» System
values
«moe» Value property 1
«moe» Value property 2
Requirement Diagram System Requirements
«requirement» System Requirement 1
«requirement» System Requirement 1.1
«requirement» System Requirement 1.2
«derive»
SysML Activity Diagram Functional Analysis
«allocate» :Subsystem 1
«allocate» :Subsystem 2
: Func 2.1
: Func 2.2
«refine»
«allocate»
«abstraction»
SysML Internal Block Diagram Subsystems Communication
p1 : Interface 1
: Subsystem 1
p2 : Interface 2
: Subsystem 2
p3 : Interface 3
«abstraction»
«derive»
Requirement Diagram Component Requirements
«requirement» Component Requirement 1
«requirement» Component Requirement 2
«satisfy»
SysML State Machine Diagram Component Behavior
SysML Activity Diagram
: Action 1
: Action 2
State 1
State 2
entry / Comp Act
State 3
SysML Internal Block Diagram Component Structure
p1 : Interface A
: Component 1
p3 : Interface B
: Component 2
: Component 3
p2 : Interface D
p4 : Interface C
SysML Parametric Diagram Component Parameters
Value property 3
«constraint» : Constraint 1
{a=b+1}
Value property 1

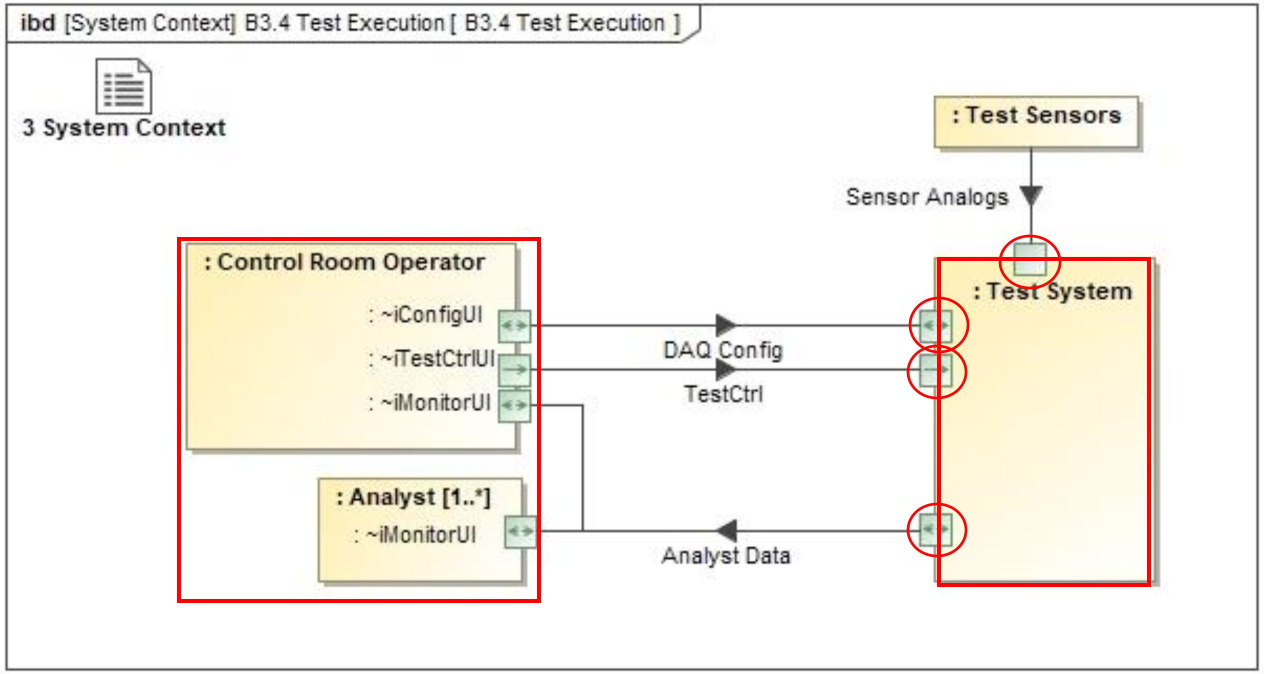
Resist Temptation



			Pillar				
			Requirements	Behavior	Structure	Parameters	
Domain	Problem	Black Box	Stakeholder Needs	Use Cases	System Context	Measurements of Effectiveness	Specialty Engineering Integrated Testing Analysis
		White Box		Functional Analysis	Logical Subsystems Communication	MoEs of Subsystems	
	Solution		System Requirements	System Behavior	System Structure	System Parameters	
			Subsystem...	...	...	...	
			Component...	...	...	...	
	Implementation		Physical Requirements	Software, Electrical, Mechanical...			

System Context

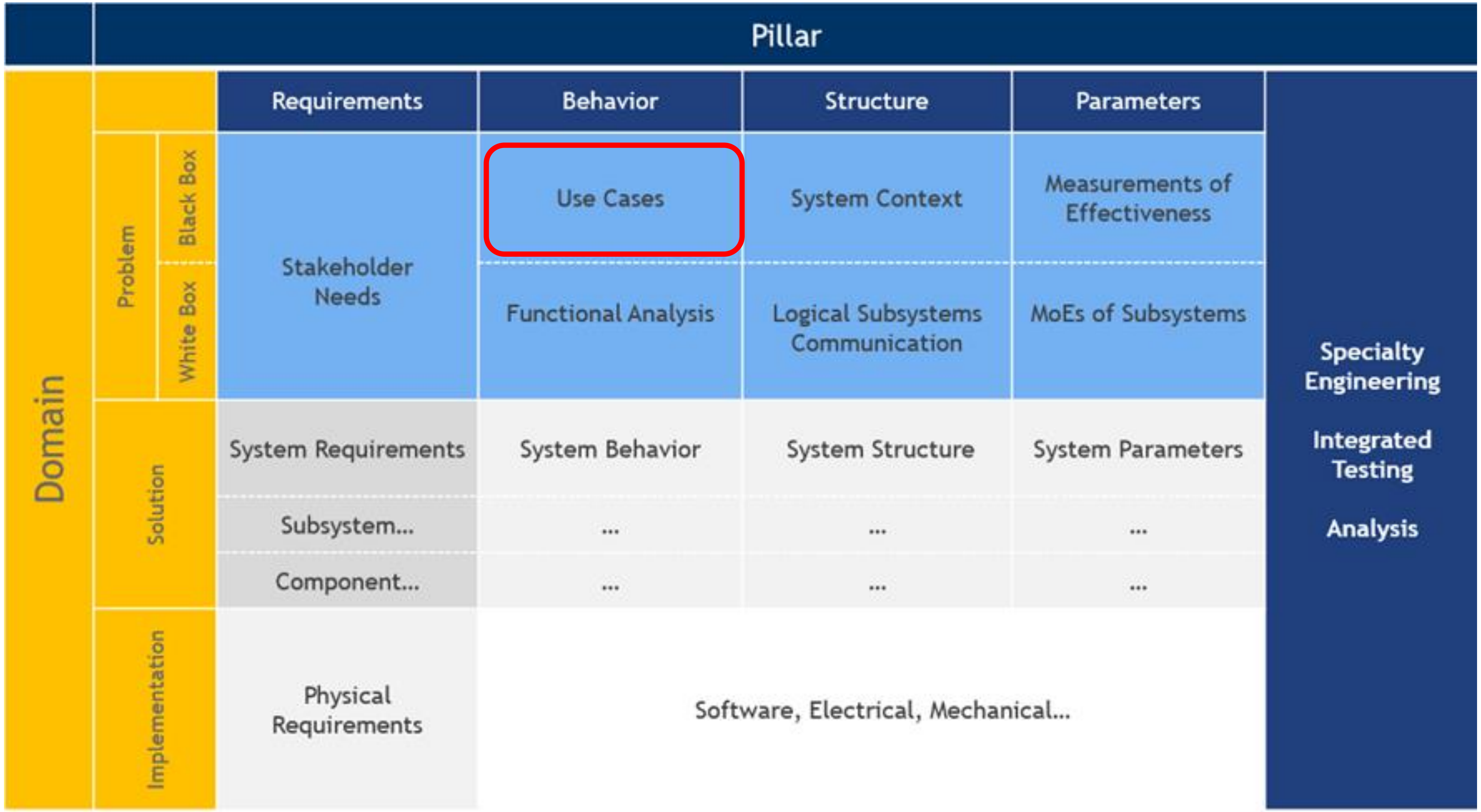
Context is an operating environment and all its external elements that interact with the system.



What is revealed?

- System Boundary
- Interfaces!
- External Actors

Use Cases

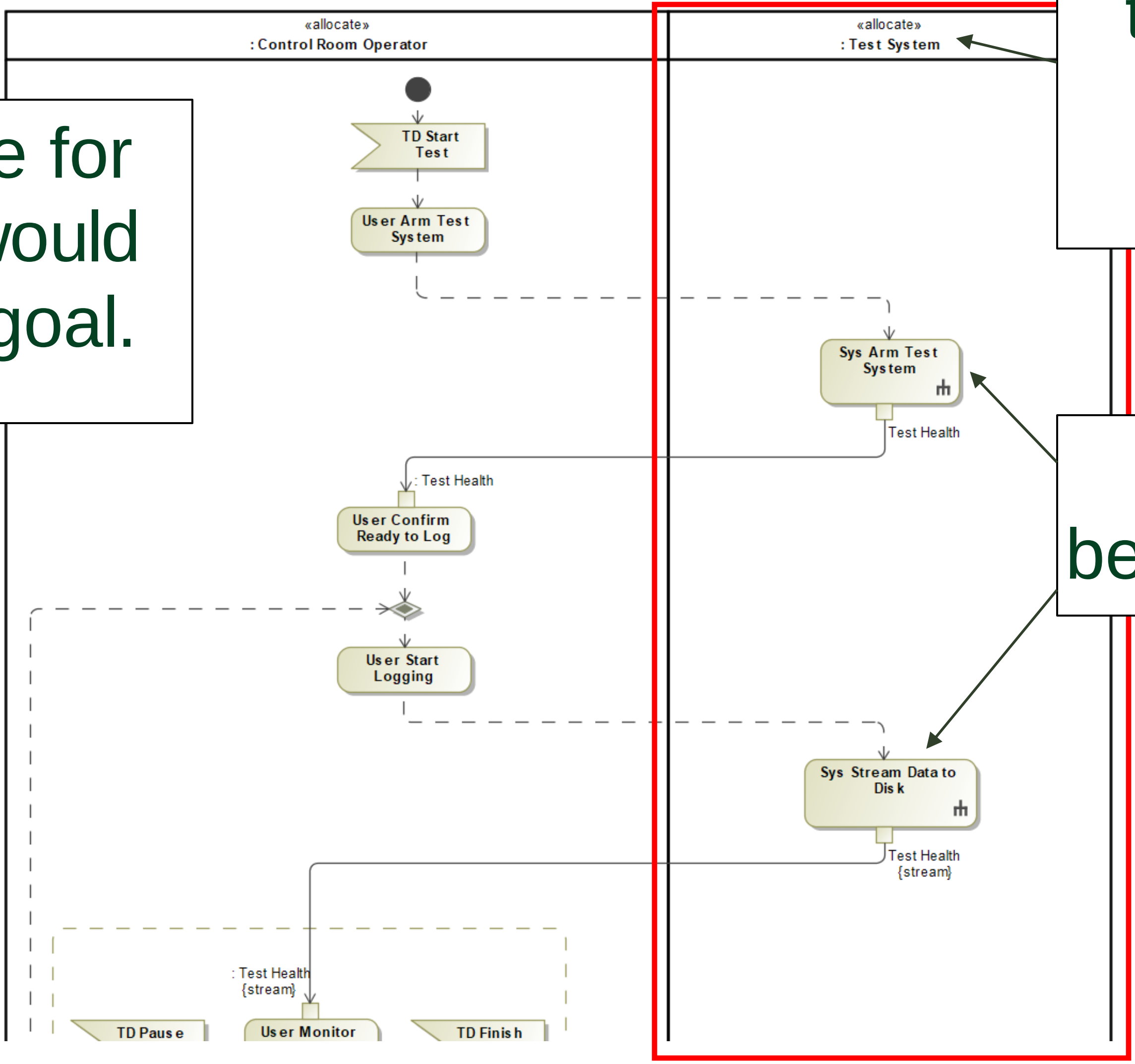


			Pillar				
Domain			Requirements	Behavior	Structure	Parameters	Specialty Engineering Integrated Testing Analysis
	Problem	Black Box	Stakeholder Needs	Use Cases	System Context	Measurements of Effectiveness	
		White Box		Functional Analysis	Logical Subsystems Communication	MoEs of Subsystems	
	Solution		System Requirements	System Behavior	System Structure	System Parameters	
			Subsystem...	...	...	...	
			Component...	...	...	...	
	Implementation		Physical Requirements	Software, Electrical, Mechanical...			

Create Workflows (Use Case Specs)

For Each Use Case:

Create narrative for how the actor would accomplish its goal.

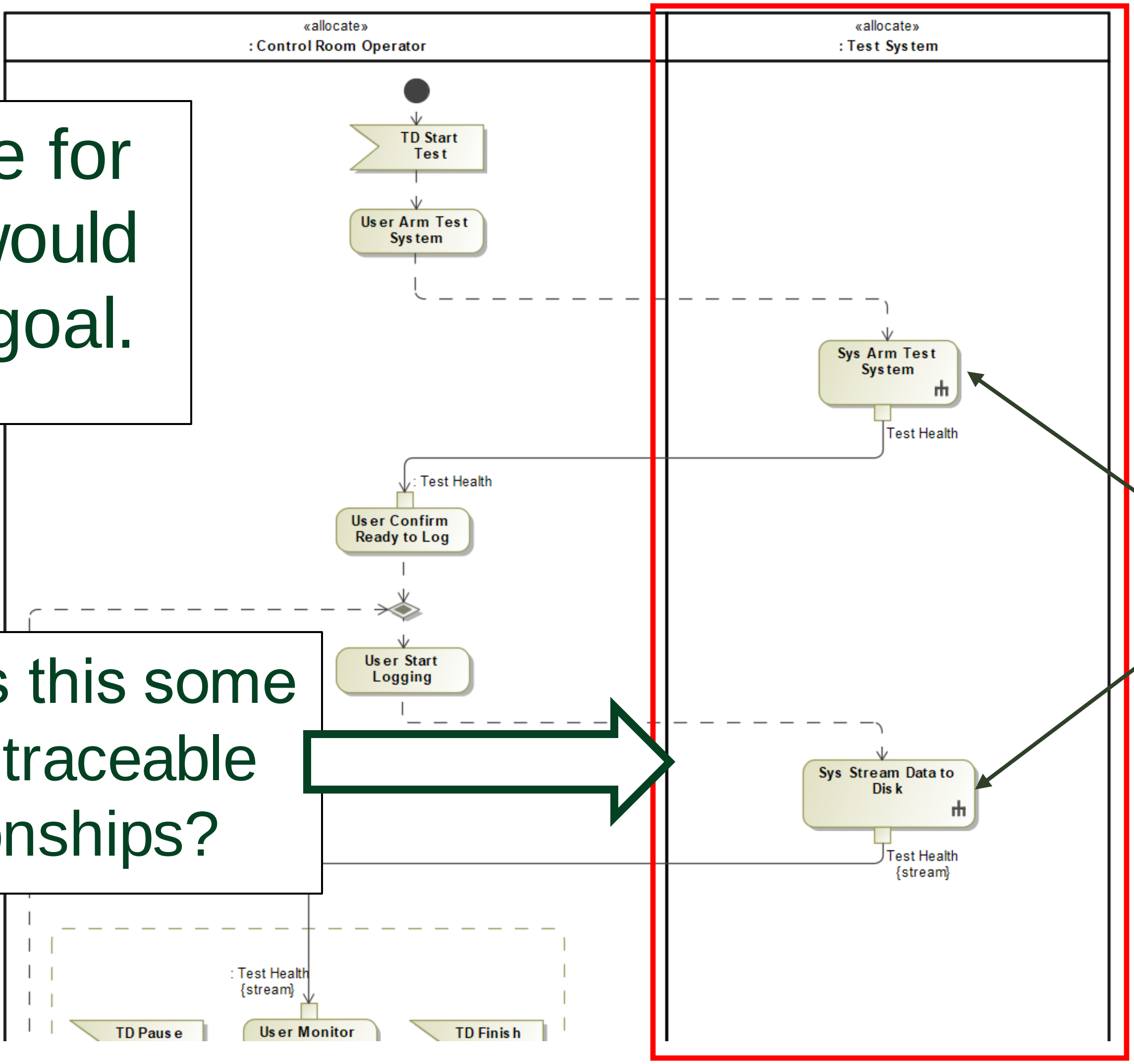


Same “Test System” that was given four interfaces in the Context diagram

Also now has two behaviors allocated to it!

Create Workflows (Use Case Specs)

For Each Use Case:



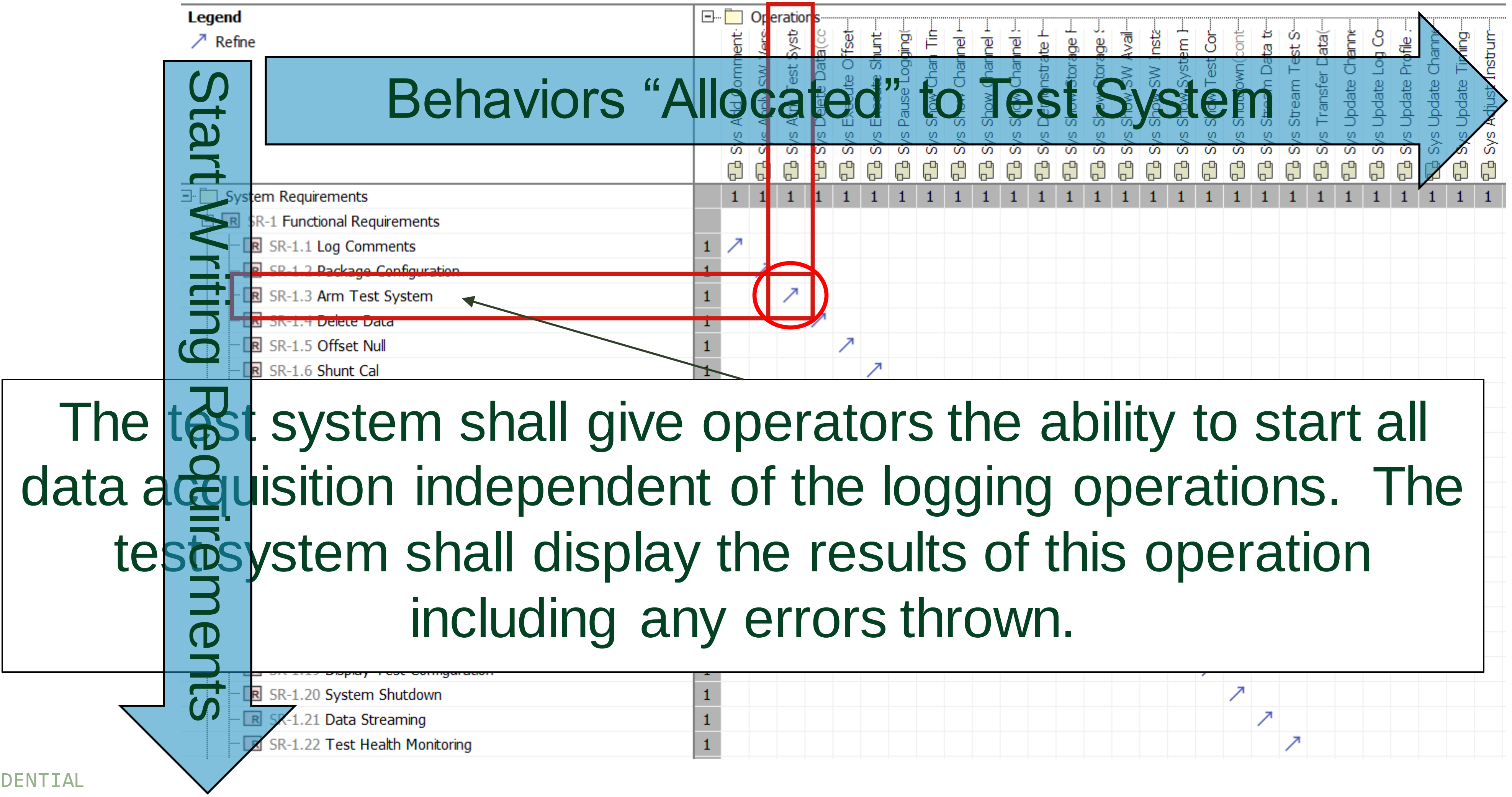
Create narrative for how the actor would accomplish its goal.

Wait… is this some kind of traceable relationships?

These smell a lot like functional requirements…

Well Let’s Make Some More!

Functional System Requirements



Legend
↗ Refine

Operations

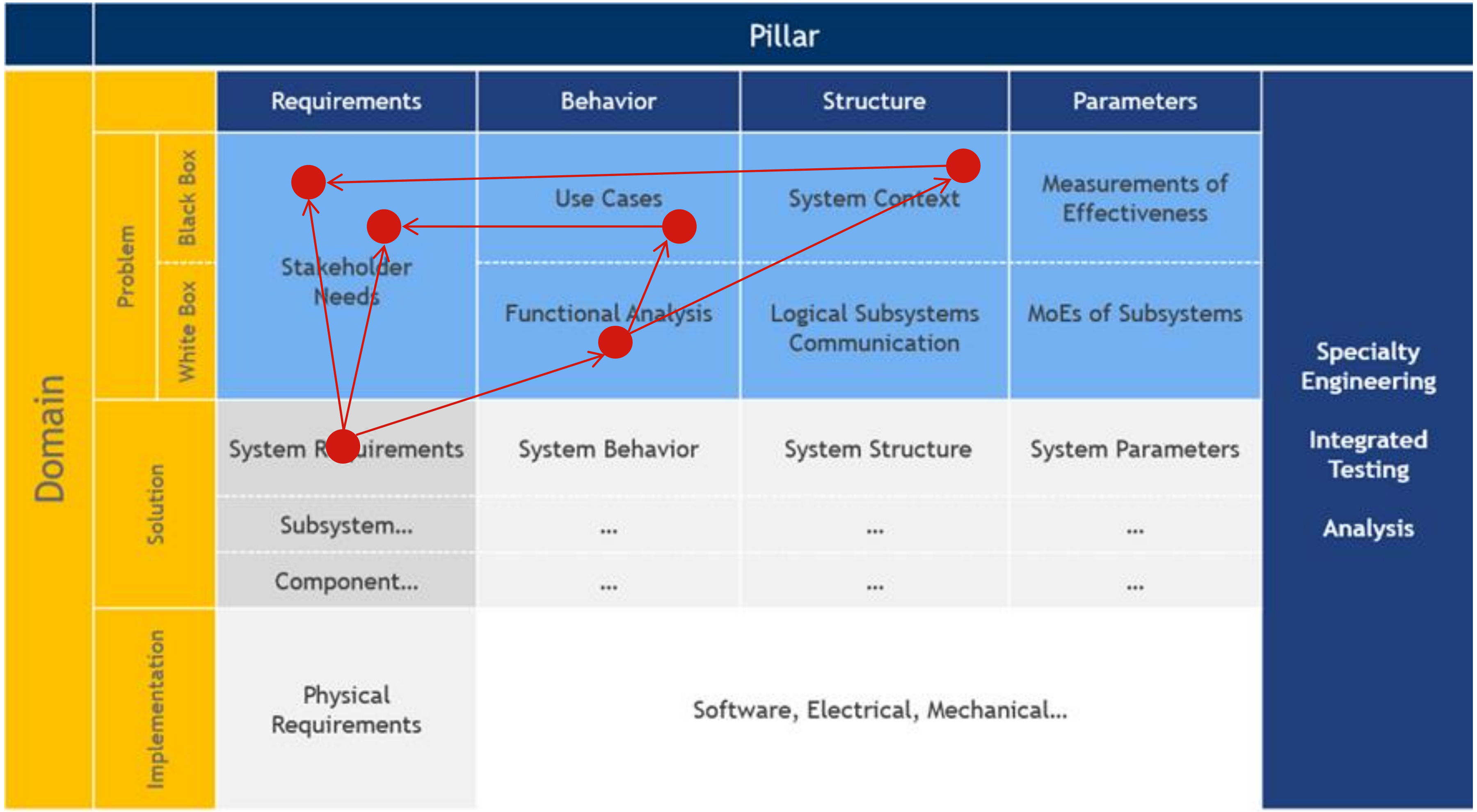
System Requirements
SR-1 Functional Requirements
SR-1.1 Log Comments 1 ↗
SR-1.2 Package Configuration 1 ↗
SR-1.3 Arm Test System 1 ↗
SR-1.4 Delete Data 1 ↗
SR-1.5 Offset Null 1 ↗
SR-1.6 Shunt Cal 1 ↗
SR-1.19 Display Test Configuration 1 ↗
SR-1.20 System Shutdown 1 ↗
SR-1.21 Data Streaming 1 ↗
SR-1.22 Test Health Monitoring 1 ↗

Behaviors “Allocated” to Test System

Start Writing Requirements

The test system shall give operators the ability to start all data acquisition independent of the logging operations. The test system shall display the results of this operation including any errors thrown.

A Traceable Network is Forming



			Pillar				
Domain			Requirements	Behavior	Structure	Parameters	Specialty Engineering Integrated Testing Analysis
	Problem	Black Box	Stakeholder Needs	Use Cases	System Context	Measurements of Effectiveness	
		White Box		Functional Analysis	Logical Subsystems Communication	MoEs of Subsystems	
	Solution		System Requirements	System Behavior	System Structure	System Parameters	
			Subsystem...	...	...	...	
			Component...	...	...	...	
	Implementation		Physical Requirements	Software, Electrical, Mechanical...			

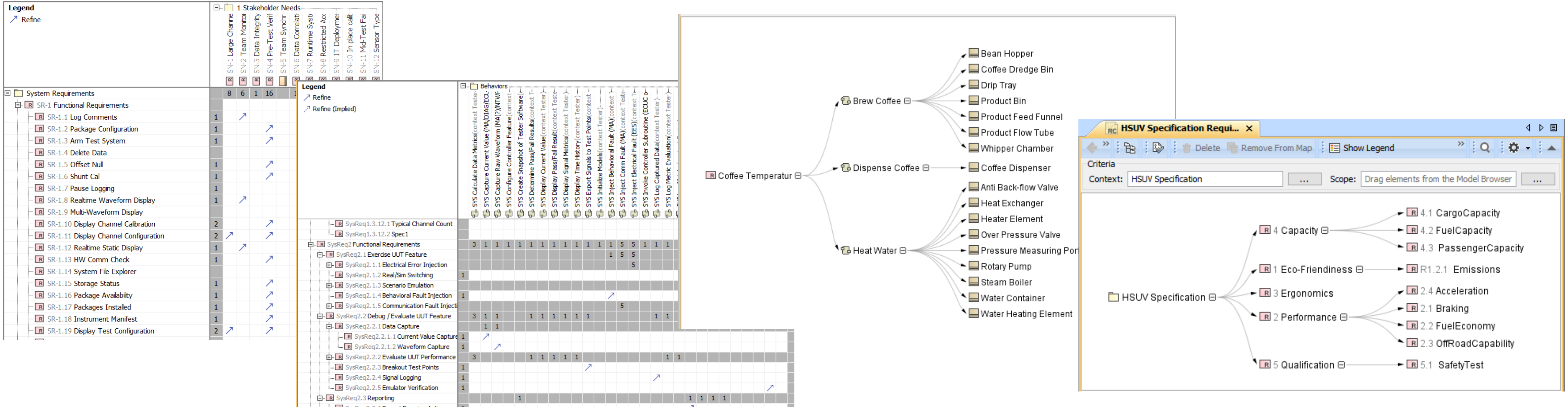
Answering the Tough Questions:

How well do the Requirements meet the need?

How well does the solution satisfy the Requirements?

If we change System requirement X, what parts of the design are impacted?

If we change component Y, what requirements are impacted?



Legend
↗ Refine

		1 Stakeholder Needs											
		SN-1 Large Channe	SN-2 Team Monitor	SN-3 Data Integrity	SN-4 Pre-Test Verif	SN-5 Team Synchr	SN-6 Data Correlati	SN-7 Runtime Syst	SN-8 Restricted Acc	SN-9 IT Deploymer	SN-10 In place calit	SN-11 Mid-Test Fai	SN-12 Sensor Type
System Requirements		8	6	1	16		1						
SR-1 Functional Requirements													
SR-1.1 Log Comments	1		↗										
SR-1.2 Package Configuration	1				↗								
SR-1.3 Arm Test System	1				↗								
SR-1.4 Delete Data													
SR-1.5 Offset Null	1				↗								
SR-1.6 Shunt Cal	1				↗								
SR-1.7 Pause Logging	1												
SR-1.8 Realtime Waveform Display	1		↗										
SR-1.9 Multi-Waveform Display													
SR-1.10 Display Channel Calibration	2				↗								
SR-1.11 Display Channel Configuration	2	↗			↗								
SR-1.12 Realtime Static Display	1		↗										
SR-1.13 HW Comm Check	1				↗								
SR-1.14 System File Explorer													
SR-1.15 Storage Status	1				↗								
SR-1.16 Package Availability	1				↗								
SR-1.17 Packages Installed	1				↗								
SR-1.18 Instrument Manifest	1				↗								
SR-1.19 Display Test Configuration	2	↗			↗								

Legend
↗ Refine
⇢ Refine (Implied)

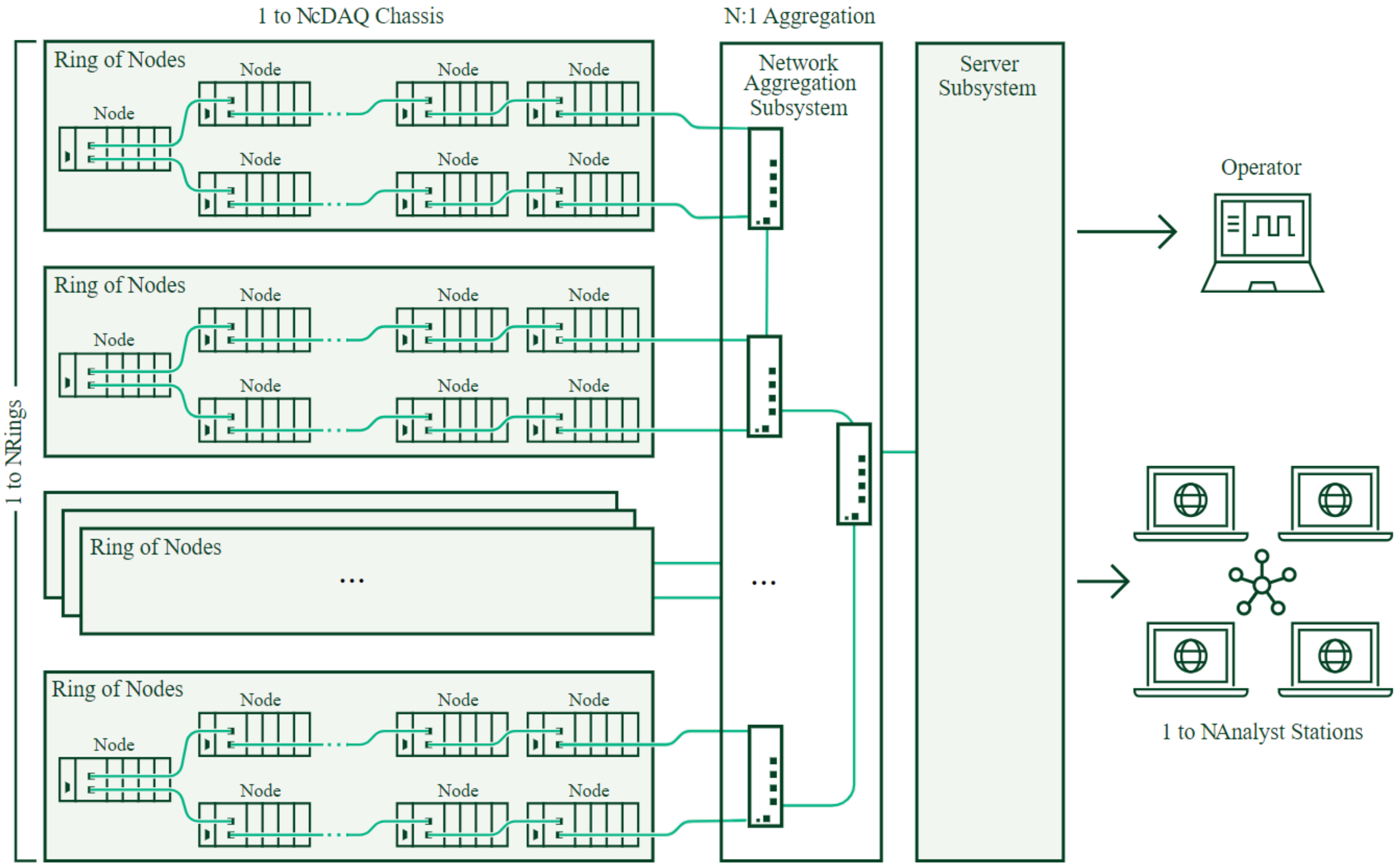
		Behaviors																											
		SYS Calculate Data Metrics(context Tester	SYS Capture Current Value (MA/DIAG/ECL	SYS Capture Raw Waveform (MA(?)/NTWF	SYS Configure Controller Feature(context	SYS Create Snapshot of Tester Software(	SYS Determine Pass/Fail Results(context T	SYS Display Current Value(context Tester)	SYS Display Pass/Fail Result(context Teste	SYS Display Signal Metrics(context Tester)	SYS Display Time History(context Tester)	SYS Export Signals to Test Points(context	SYS Initialize Models(context Tester)	SYS Inject Behavioral Fault (MA)(context T	SYS Inject Comm Fault (MA)(context Teste	SYS Inject Electrical Fault (EES)(context T	SYS Invoke Controller Subroutine (ECUC o	SYS Log Captured Data(context Tester)	SYS Log Metric Evaluation(context Tester)										
SysReq1.3.12.1 Typical Channel Count																													
SysReq1.3.12.2 Spec1																													
SysReq2 Functional Requirements		3	1	1	1	1	1	1	1	1	1	1	1	1	5	5	1	1	1										
SysReq2.1 Exercise UUT Feature														1	5	5													
SysReq2.1.1 Electrical Error Injection																5													
SysReq2.1.2 Real/Sim Switching	1																												
SysReq2.1.3 Scenario Emulation																													
SysReq2.1.4 Behavioral Fault Injection	1													↗															
SysReq2.1.5 Communication Fault Injecti															5														
SysReq2.2 Debug / Evaluate UUT Feature		3	1	1			1	1	1	1	1	1						1	1										
SysReq2.2.1 Data Capture			1	1																									
SysReq2.2.1.1 Current Value Capture	1		↗																										
SysReq2.2.1.2 Waveform Capture	1			↗																									
SysReq2.2.2 Evaluate UUT Performance		3					1	1	1	1	1								1	1									
SysReq2.2.3 Breakout Test Points	1											↗																	
SysReq2.2.4 Signal Logging	1																	↗											
SysReq2.2.5 Emulator Verification	1																											↗	
SysReq2.3 Reporting						1															1	1	1	1					

Context: HSUV Specification

Scope: Drag elements from the Model Browser

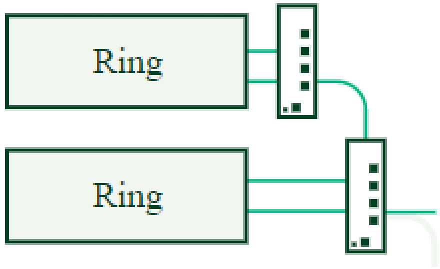
1. Application Space
2. Problem to Solve
3. Using MBSE to Gather Industry Requirements
4. Hardware Architecture
5. Software Suite
6. Validating the Architecture
7. Project Deliverables
8. Lessons Learned

Hardware Architecture

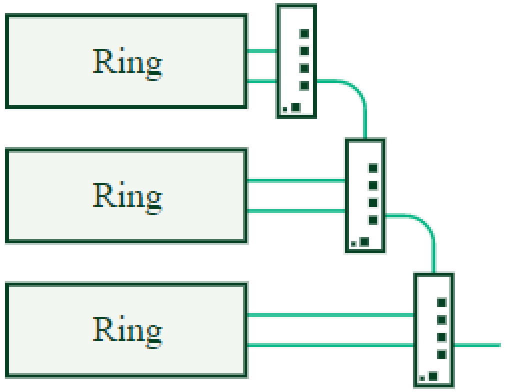


Ring Pair Aggregation

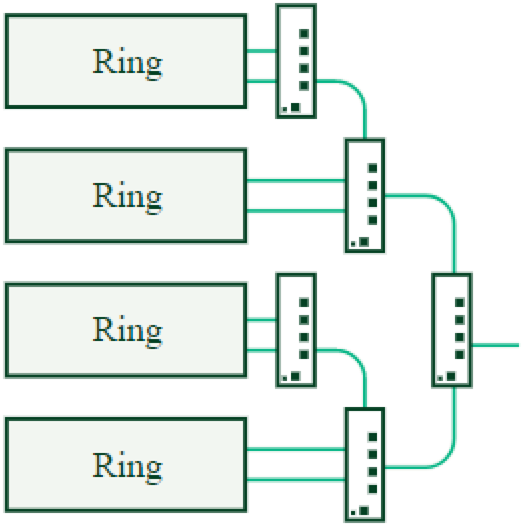
2 Rings



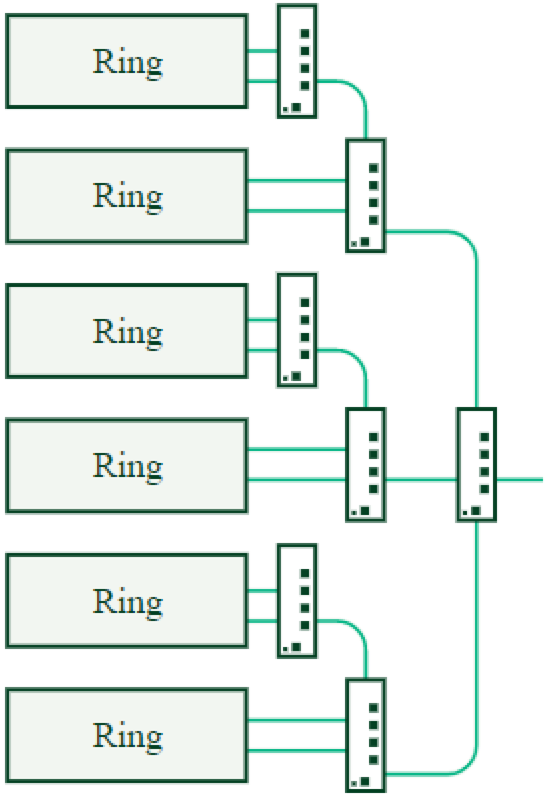
3 Rings



4 Rings

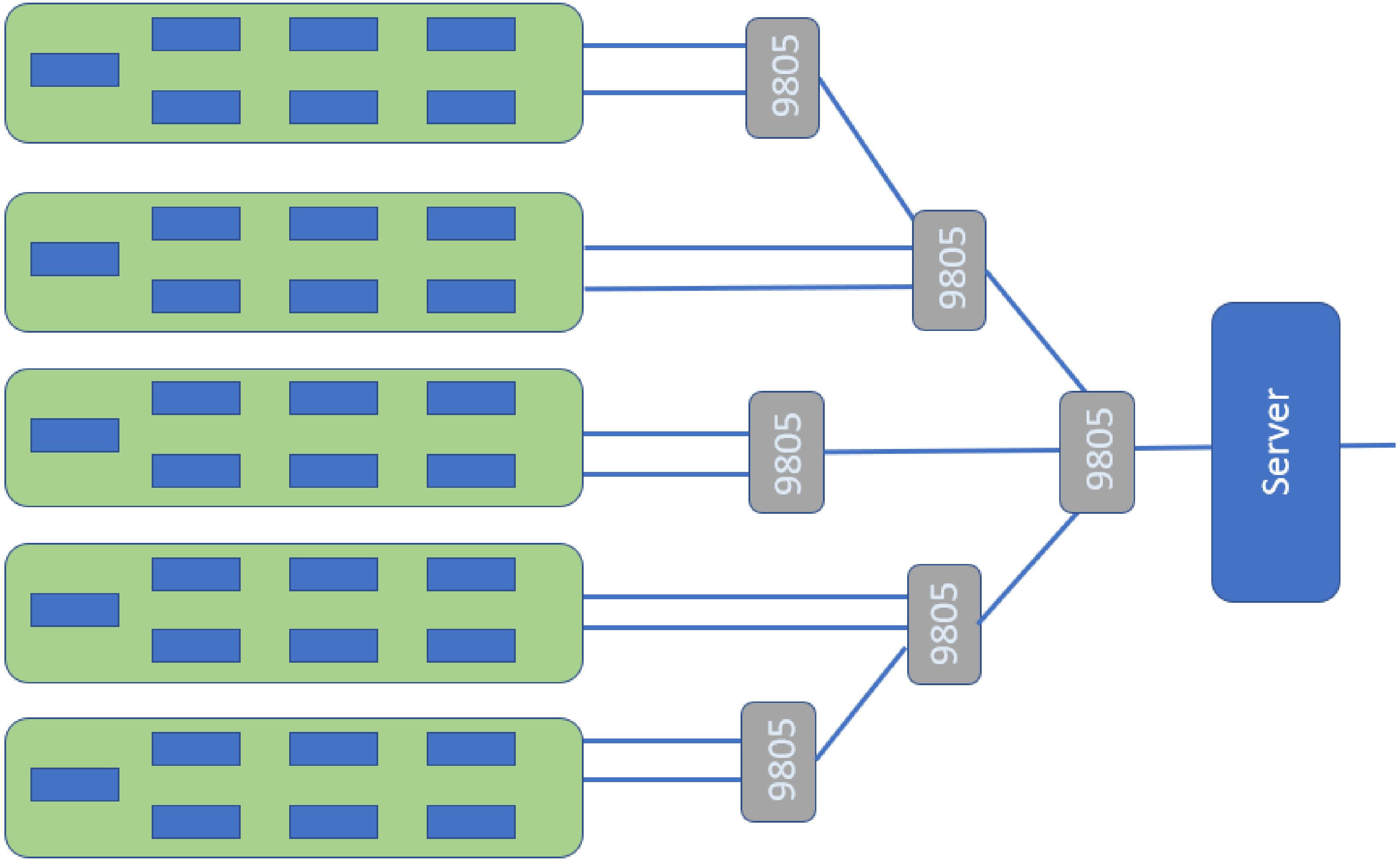


6 Rings

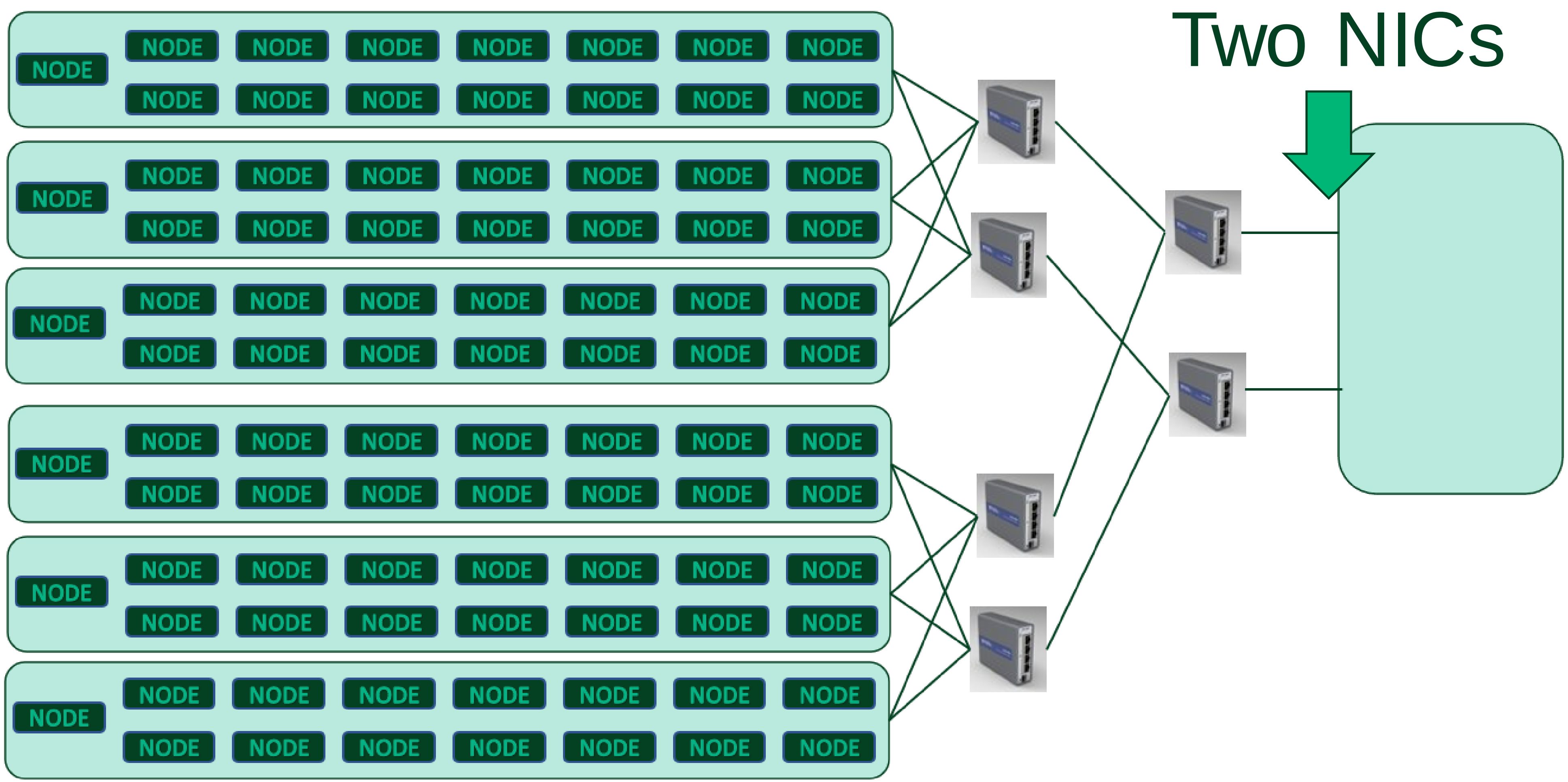


Redundancy – Example System

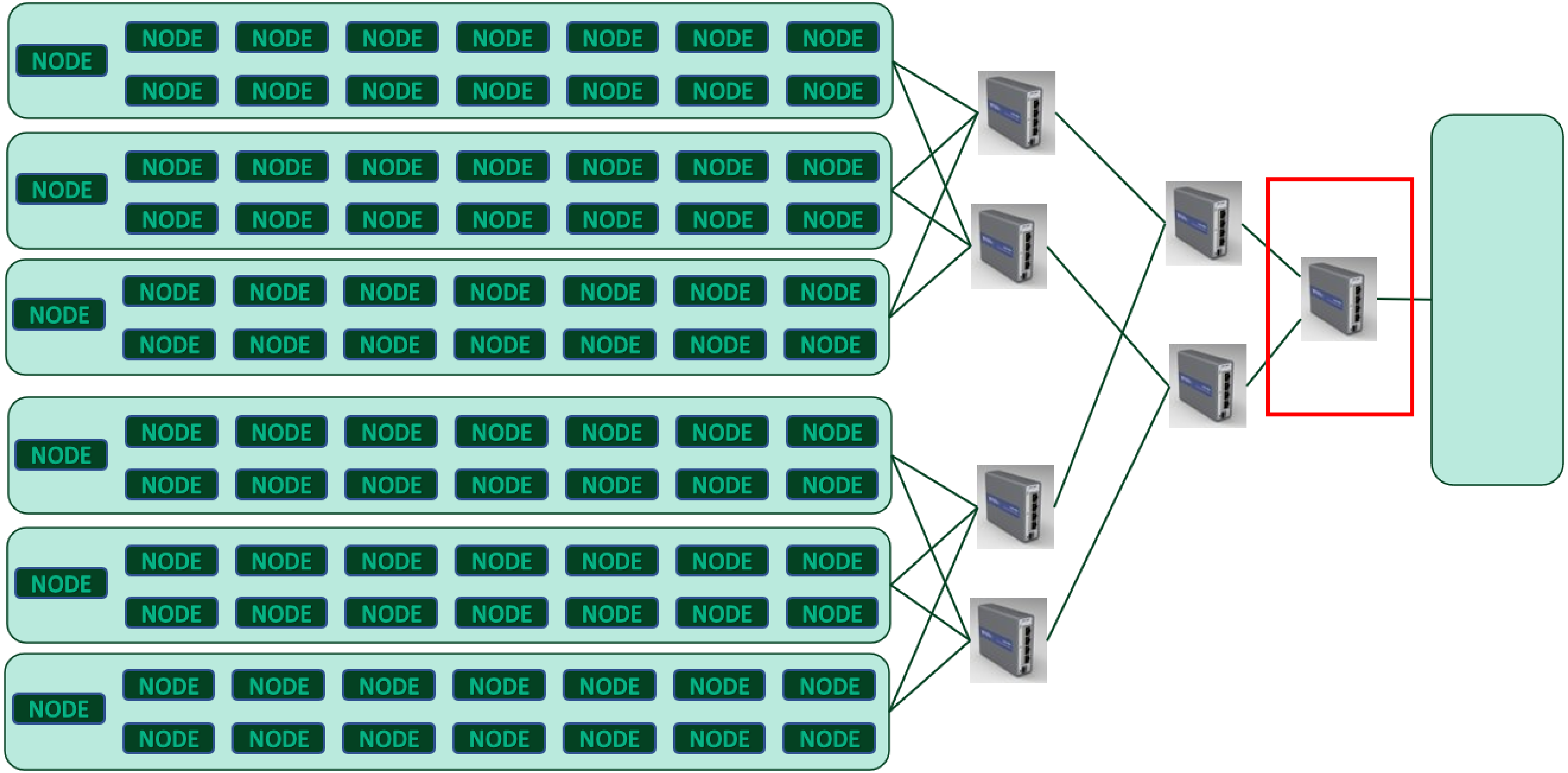
How many single points of failure?



Redundancy – Full Network Redundancy

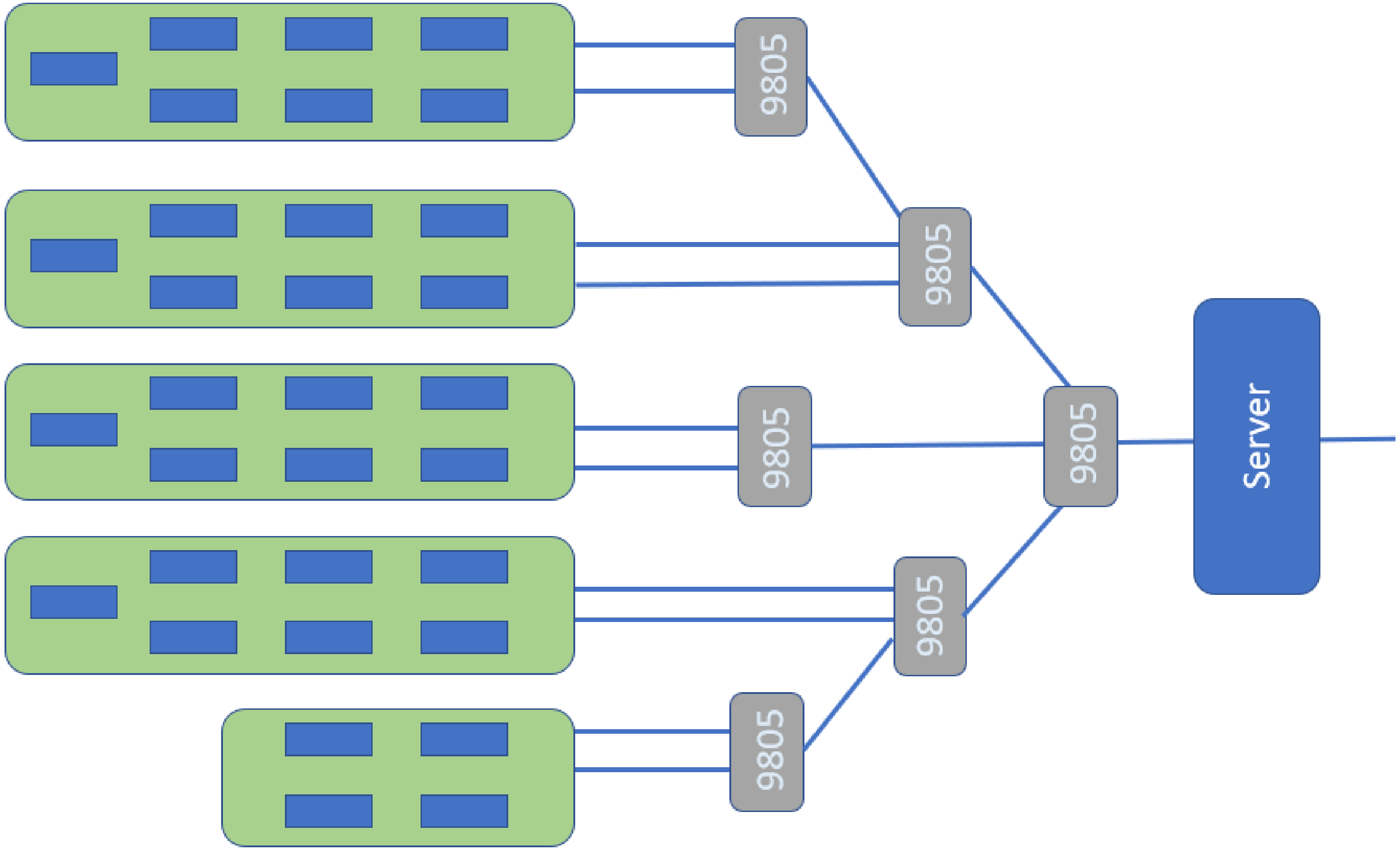


Redundancy – HybridRedundancy



Max Hops

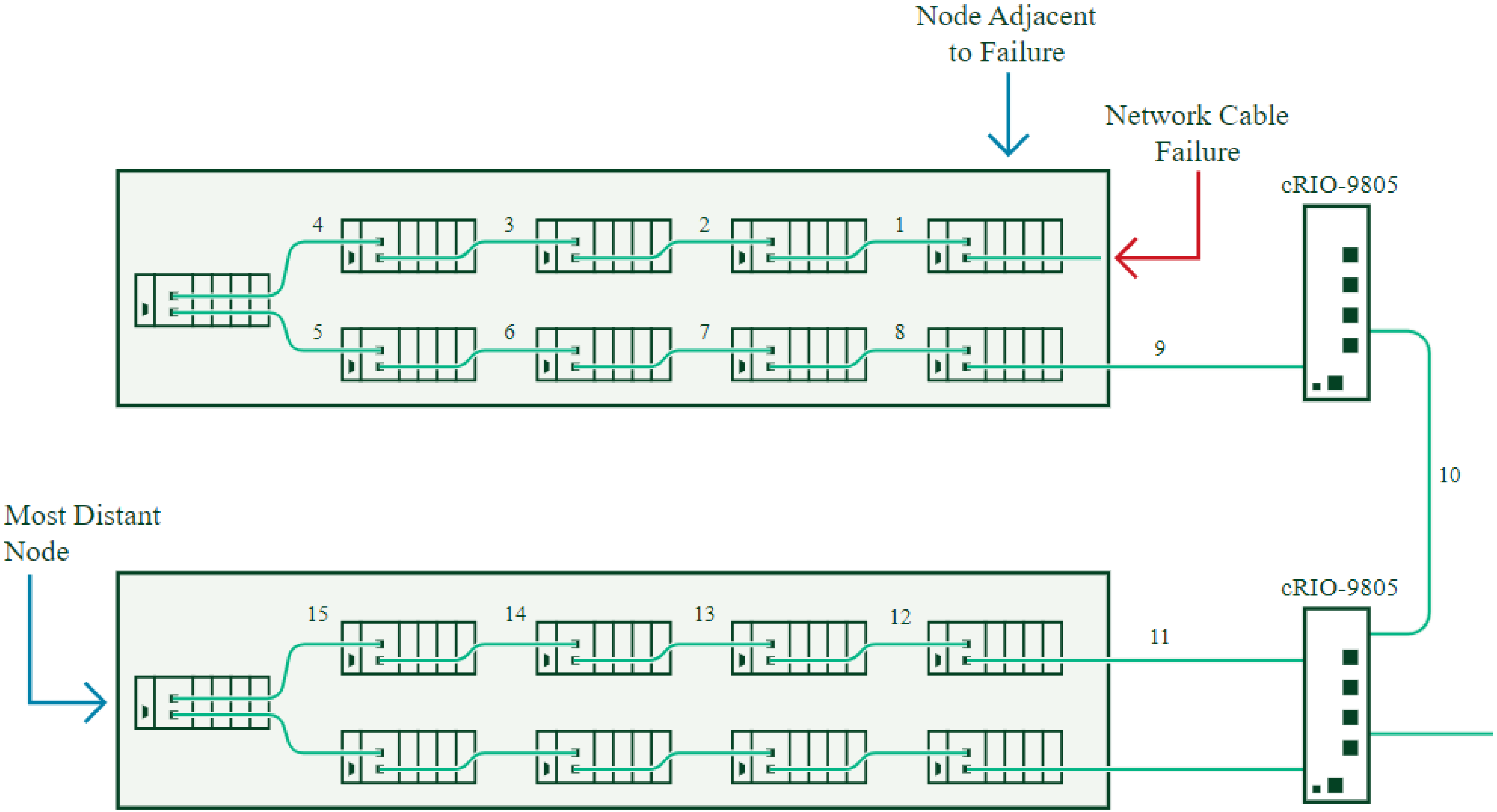
No node can be more than 15 hops from the “root” node



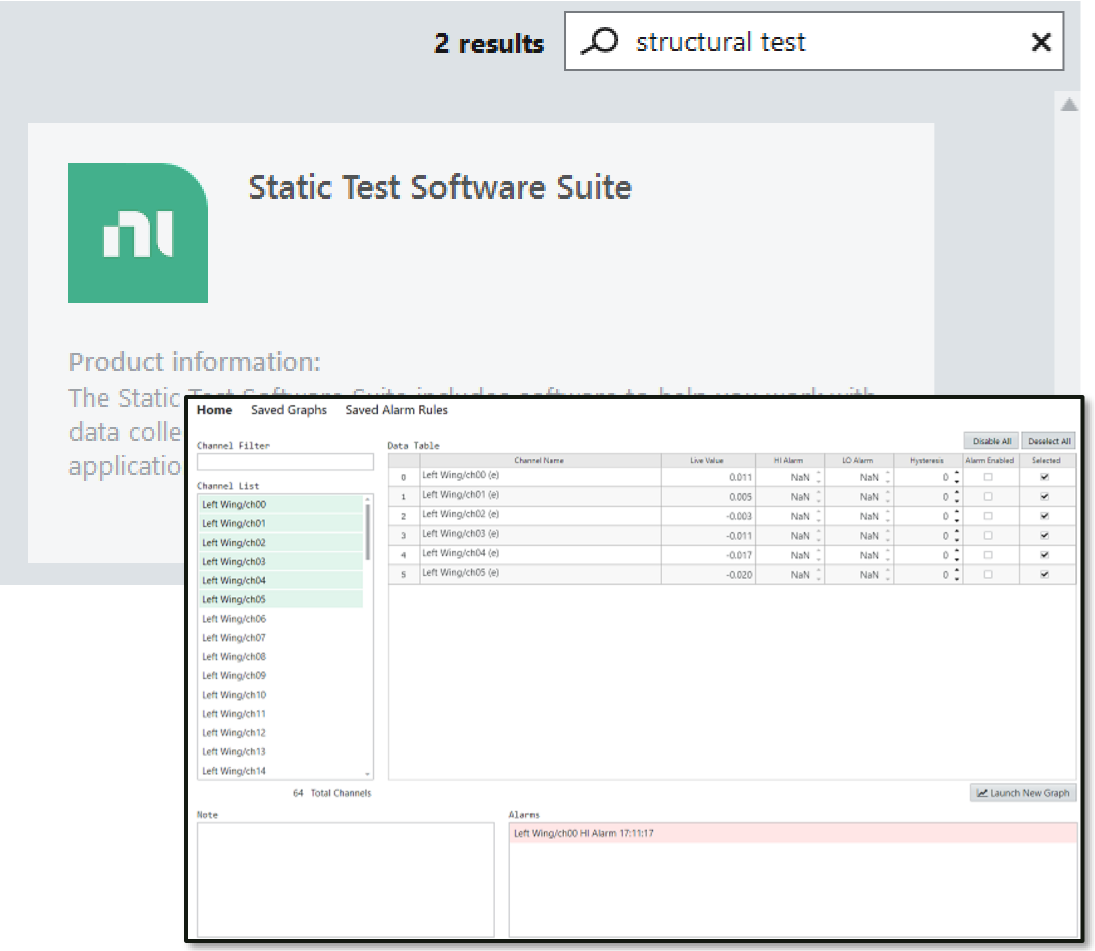
# Nodes	Nodes per Ring
1 – 15	15
16 - 18	9
18 - 42	7

Max Hops

Worst case: Ring of N nodes reconfigures into daisy chain on N nodes



Software Suite



2 results

structural test ✕

Static Test Software Suite

Product information:

The Static Test Software Suite includes software to help you work with

data colle

applicatio

Home Saved Graphs Saved Alarm Rules

Channel Filter

Channel List

Left Wing/ch00
Left Wing/ch01
Left Wing/ch02
Left Wing/ch03
Left Wing/ch04
Left Wing/ch05
Left Wing/ch06
Left Wing/ch07
Left Wing/ch08
Left Wing/ch09
Left Wing/ch10
Left Wing/ch11
Left Wing/ch12
Left Wing/ch13
Left Wing/ch14

64 Total Channels

Data Table

Disable All | Deselect All

	Channel Name	Live Value	HI Alarm	LO Alarm	Hysteresis	Alarm Enabled	Selected
0	Left Wing/ch00 (e)	0.011	NaN	NaN	0	□	☑
1	Left Wing/ch01 (e)	0.005	NaN	NaN	0	□	☑
2	Left Wing/ch02 (e)	-0.003	NaN	NaN	0	□	☑
3	Left Wing/ch03 (e)	-0.011	NaN	NaN	0	□	☑
4	Left Wing/ch04 (e)	-0.017	NaN	NaN	0	□	☑
5	Left Wing/ch05 (e)	-0.020	NaN	NaN	0	□	☑

Launch New Graph

Note

Alarms

Left Wing/ch00 HI Alarm 17:11:17

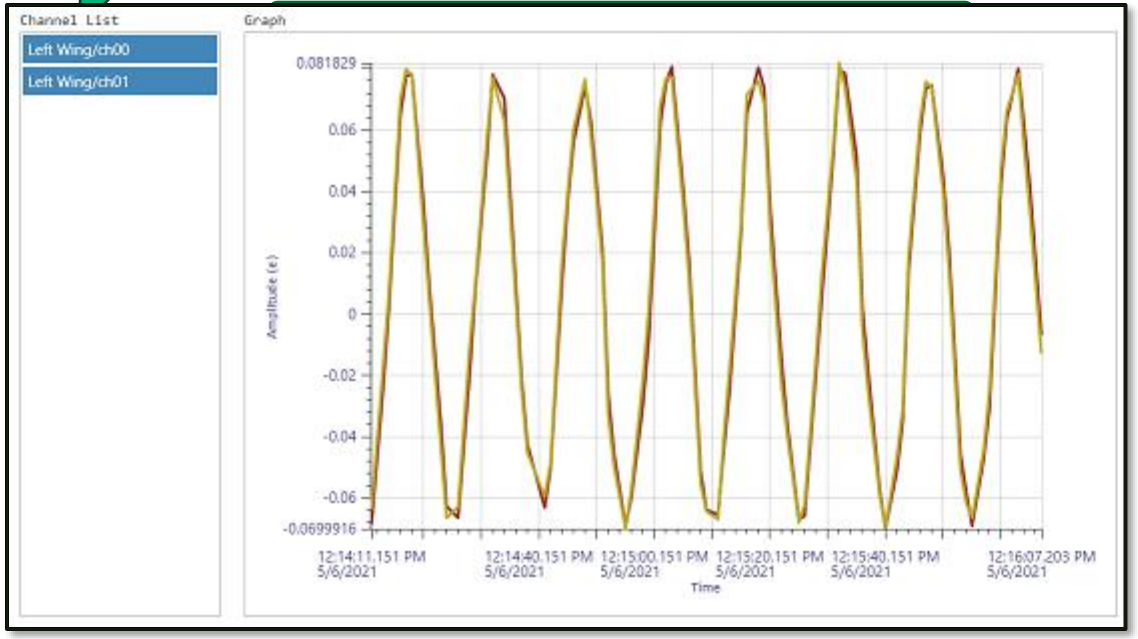
System Software Stack

NI DAQmx Firmware

NI DAQmx Driver

FlexLogger

FlexLogger Plugin



1. Application Space
2. Problem to Solve
3. Using MBSE to Gather Industry Requirements
4. Hardware Architecture
5. Software Suite
6. Validating the Architecture
7. Project Deliverables
8. Lessons Learned

Validating the Architecture

2048 Channels

Two Weeks Continuous Logging @ 100 Hz

- ~50% CPU Utilization
- ~25% Memory Utilization

Parameterized Testing:

- 448 Channels – 10kS/s
- 1160 Channels – 10kS/s
- 2048 Channels – 3kS/s

Shunt Cal / Offset Null

- 1944 Strain Channels - ~5min

1. Application Space
2. Problem to Solve
3. Using MBSE to Gather Industry Requirements
4. Hardware Architecture
5. Software Suite
6. Validating the Architecture
7. Project Deliverables
8. Lessons Learned

Project Deliverables

- Designing Systems Using the Static Structural Test Reference Architecture
  https://www.ni.com/docs/en-US/bundle/designing-static-structural-test/page/designing.html

- Commissioning a System from the Static Structural Test Reference Architecture
  https://www.ni.com/docs/en-US/bundle/commissioning-static-structural-test/page/commissioning.html

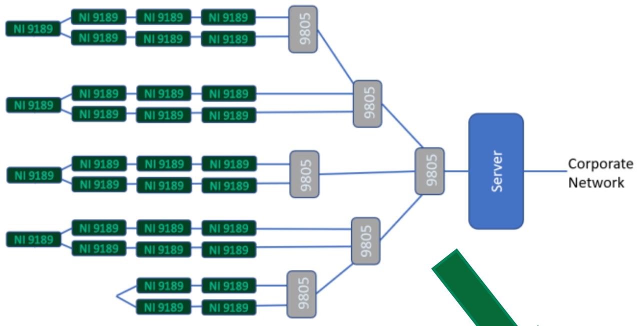
- Benchmarking a System Built from the Static Structural Test Reference Architecture
  https://www.ni.com/docs/en-US/bundle/benchmarking-static-structural-test/page/benchmarking.html

Project Deliverables

1944 Strain
32 Voltage
64 Temperature

System Design Guide

System Schematic and BOM



User Manual / Installation Guide

ADG Test System

System Benchmarks and Performance

Validated and *Versioned* Installer

Lessons Learned

You need a GOOD set of common industry requirements

Model got stale as we went to execution (no single source of truth)

- Cameo is a good tool for tracking design against requirements
- Azdo is a good tool for tracking work against requirements
- Methodology was the most valuable part of the whole exercise

SysML doesn’t have native language support for Variant Modeling

TSN rings can’t straddle switches, limiting our redundancy capabilities

FlexLogger can perform 2000 channel tests over two weeks

CONNECT