Agile for System Development

Foundations, Good Ideas and Conversation Starters

Kelly Weyrauch
Agile Quality Systems LLC

Kelly Weyrauch
Agile Quality Systems, LLC

- 23 years medical device software & system development
- Author/owner/user of SOPs & Work Instructions for software and systems development
- ASQ Certified Quality Auditor, focusing on Design Controls, Risk Management, Complaint Handling
- 14 years Agile practitioner, Trainer, Agile Transformation Coach, Certified Scrum Master, Certified SAFe Program Consultant
- Lead author of AAMI TIR45 “Guidance on the use of AGILE practices in the development of medical device software” - Lead instructor for AAMI course on TIR45
- Adjunct Instructor for Design Control courses in St. Cloud State’s Master of Technology Quality program
- Consultant to large & small medical device organizations

Agile is Not Just for Software

- Scrum is used in many product-development contexts
- Agile Scaling concepts and frameworks address big system development
  - And integration of software teams using Agile with other development teams that may / may not be
- Scaled Agile Framework (SAFe)
  - Version 4.0 puts more emphasis on development of systems, aligning with Systems Engineering principles
  - For “large-scale, multidisciplinary software and cyber-physical systems”

Software development is not just Software Engineering

- Embedded software connects with products/ subsystems created by other engineering disciplines
- Big “software-only” systems need discipline of Systems Engineering

- “The Systems Engineering Engine”
  - NASA Systems Engineering Handbook provides a model of activities for developing big systems

Agile + Systems Engineering

- Agile (SAFe + Scrum + XP + Kanban + …)

+

- Systems Engineering concepts

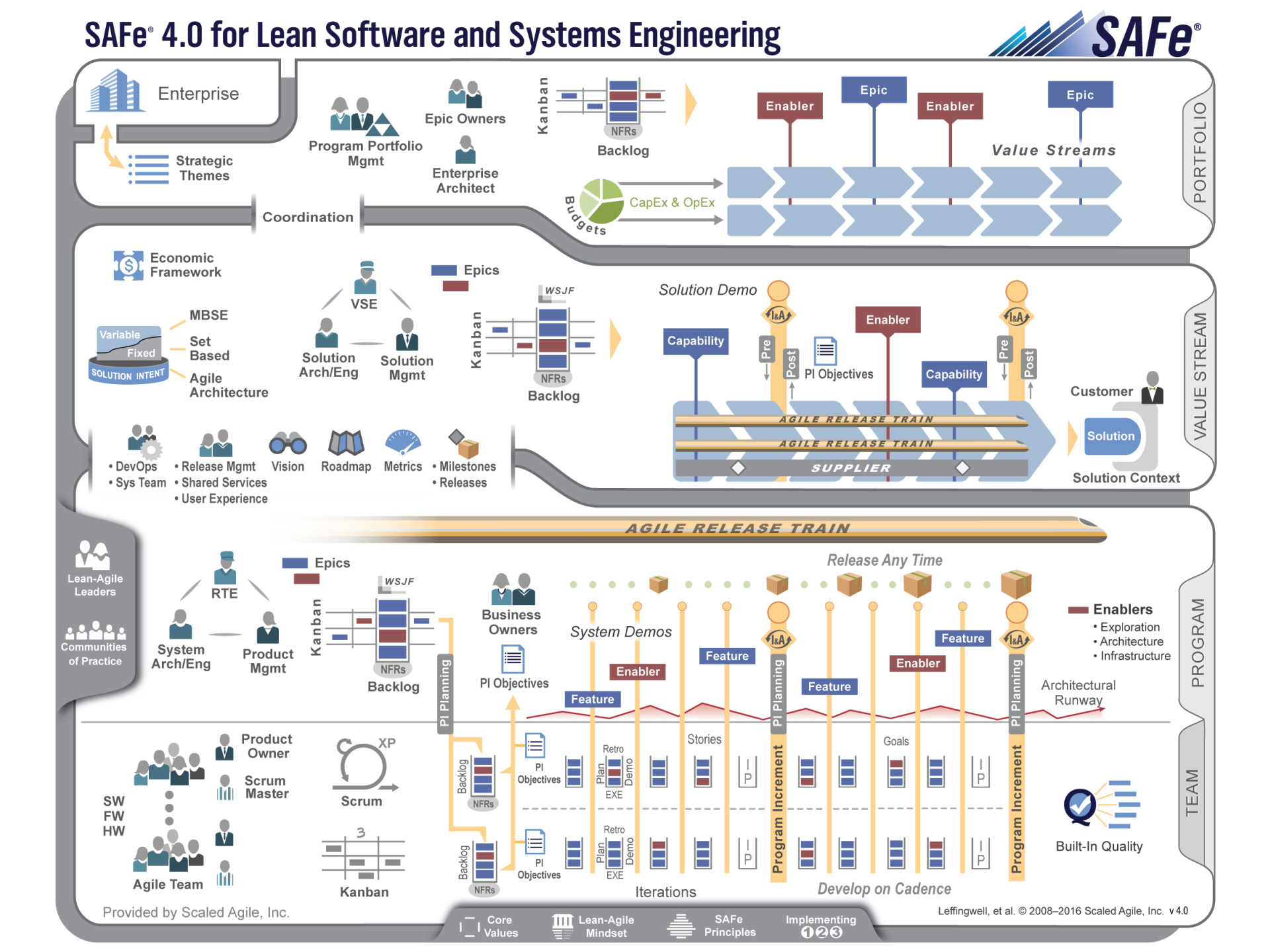
=

- A really good thing

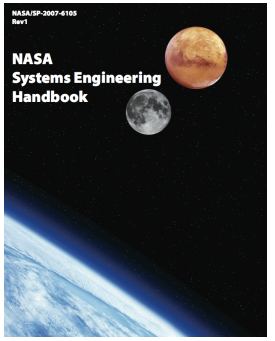
SAFe® content from Scaled Agile Inc

- The “Big Picture”
  - http://scaledagileframework.com
- “SAFe 4.0 in 8 Pictures”
  - http://scaledagileframework.com/videos-and-presentations/

SAFe® 4.0 for Lean Software and Systems Engineering



NASA SE Handbook



Requirements flow down from level above

Realized products to level above

SYSTEM DESIGN PROCESSES

Requirements Definition Processes
1. Stakeholder Expectations Definition
2. Technical Requirements Definition

Technical Solution Definition Processes
3. Logical Decomposition
4. Design Solution Definition

TECHNICAL MANAGEMENT PROCESSES

Technical Planning Process
10. Technical Planning

Technical Control Processes
11. Requirements Management
12. Interface Management
13. Technical Risk Management
14. Configuration Management
15. Technical Data Management

Technical Assessment Process
16. Technical Assessment

Technical Decision Analysis Process
17. Decision Analysis

PRODUCT REALIZATION PROCESSES

Product Transition Process
9. Product Transition

Evaluation Processes
7. Product Verification
8. Product Validation

Design Realization Processes
5. Product Implementation
6. Product Integration

Requirements flow down to level below

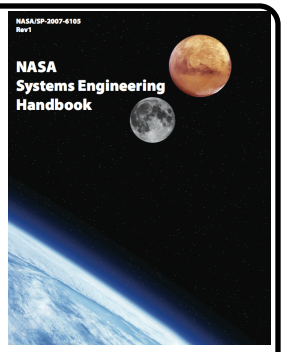
System design processes applied to each work breakdown structure model down and across system structure

Realized products from level below

Product realization processes applied to each product up and across system structure

Figure 2.1-1 The systems engineering engine

The Systems Engineering Engine



System Development Processes

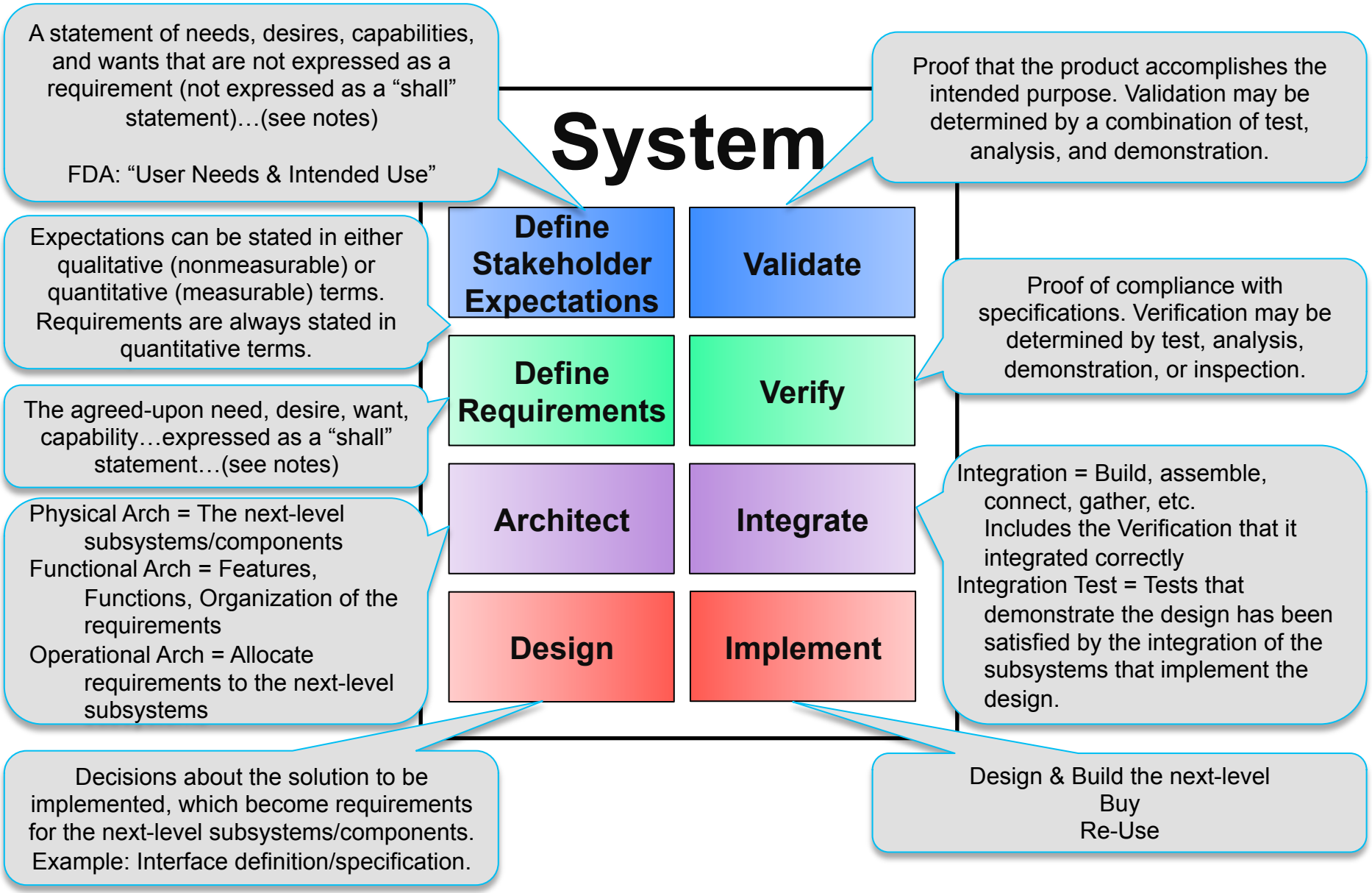
System Design	Product Realization
Define Stakeholder Expectations	Validate
Define Requirements	Verify
Architect	Integrate
Design	Implement

Technical Management Processes

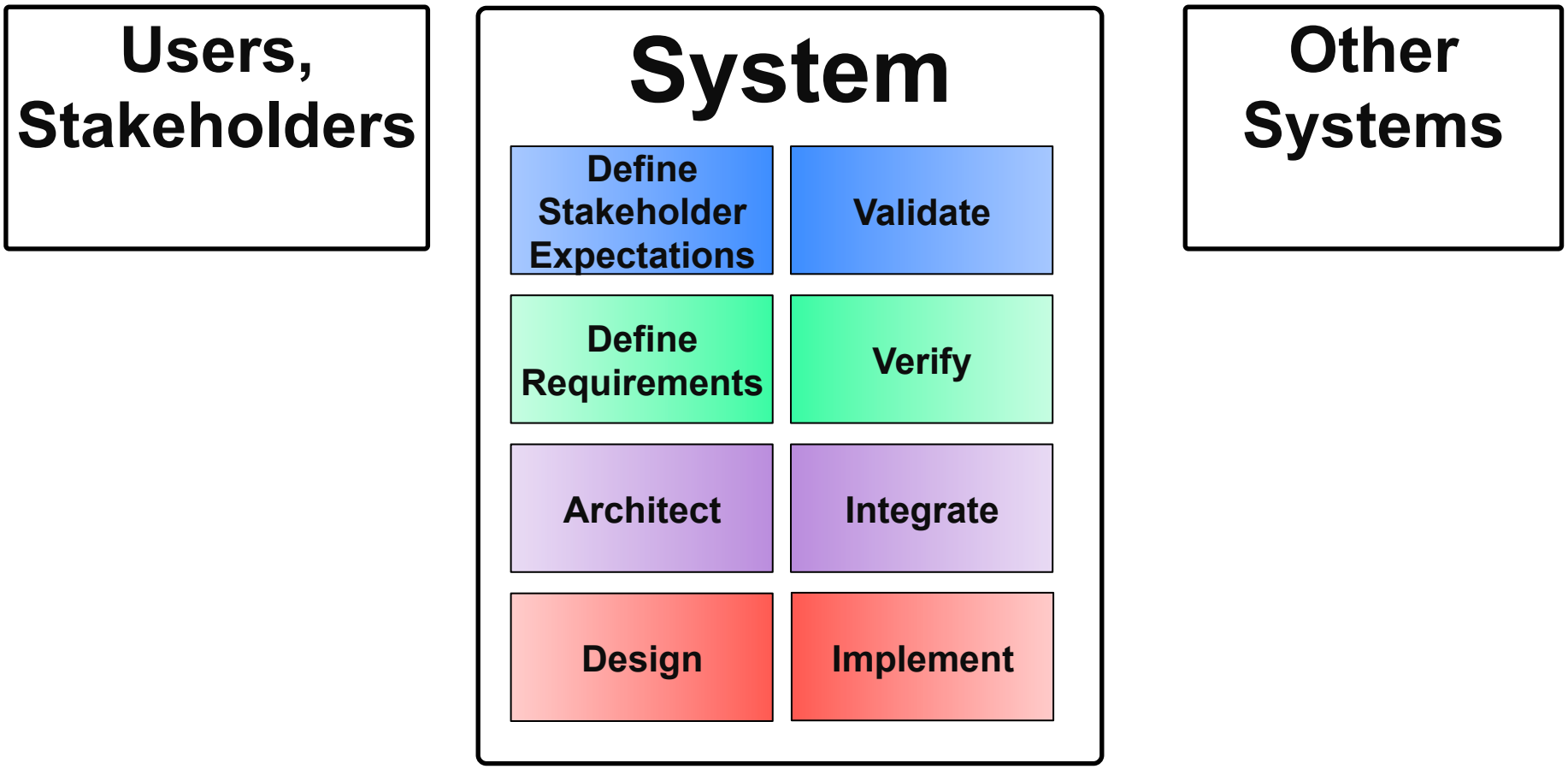
- Technical Planning
- Requirements Management
- Interface Management
- Risk Management
- Configuration Management
- Data Management

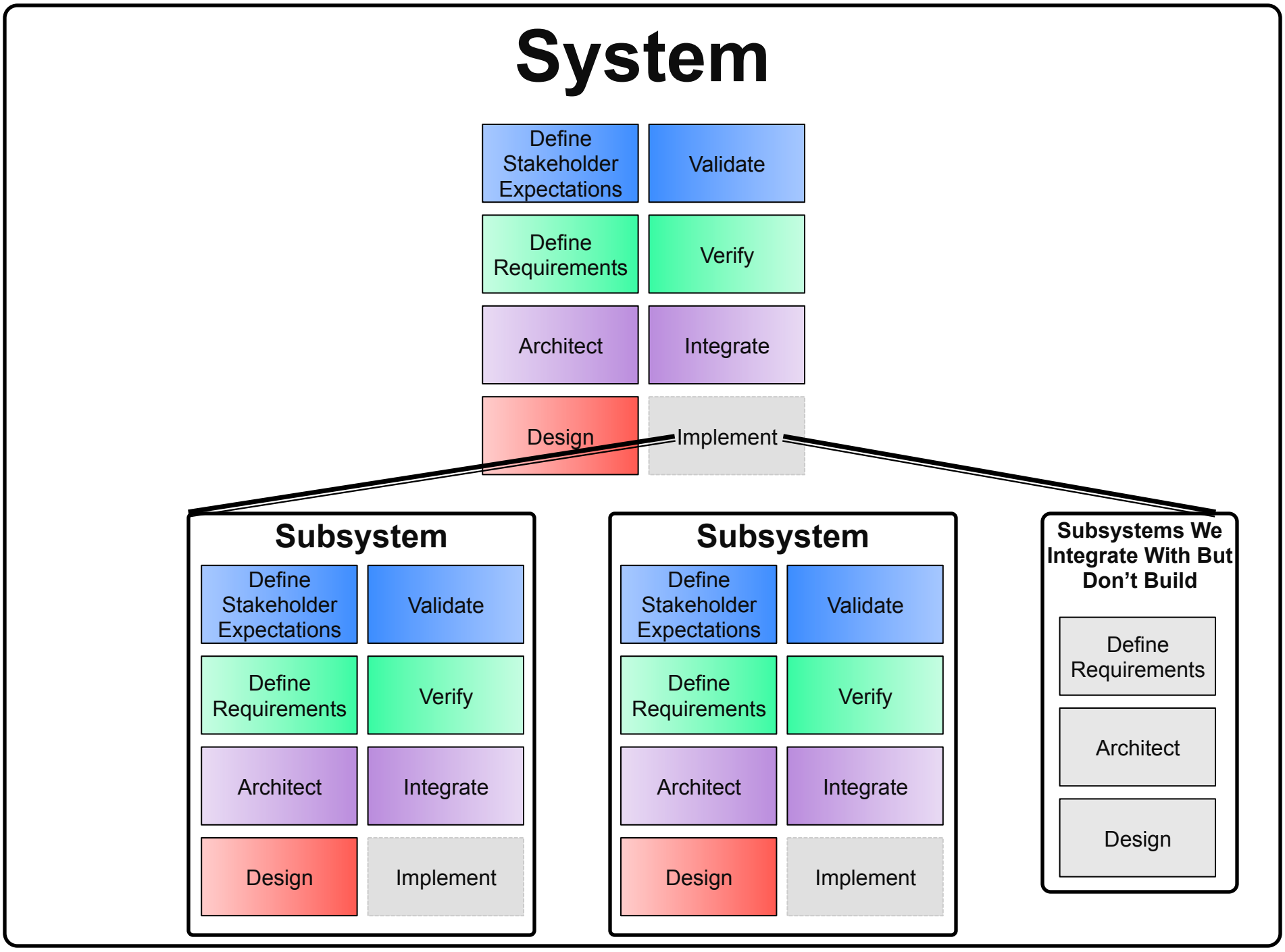
Adapted from: NASA Systems Engineering Handbook, NASA/SP-2007-6105

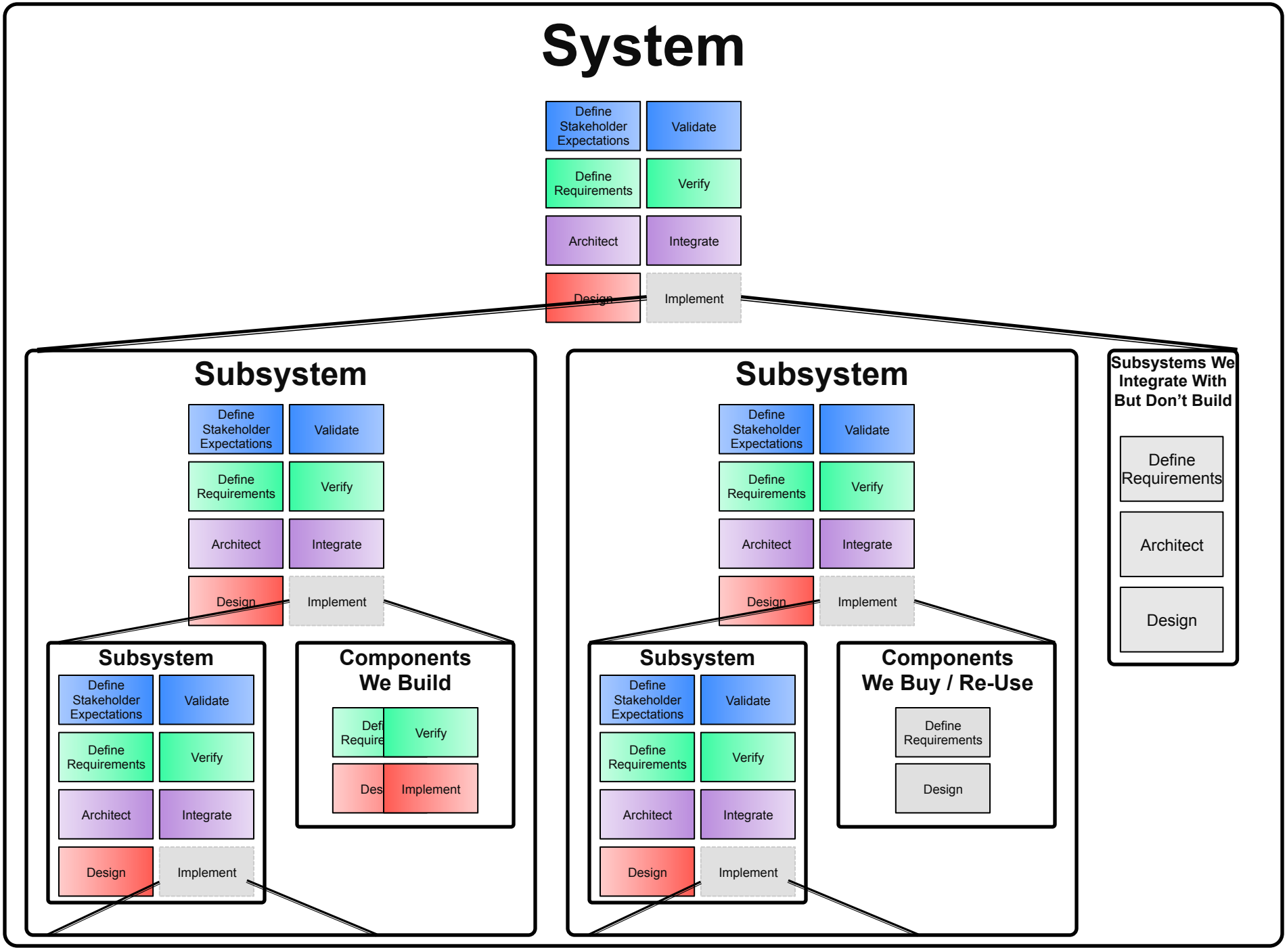
The Systems Engineering Engine



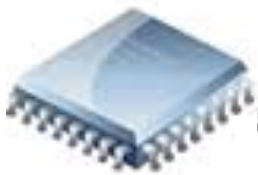
Systems Live Within A Context



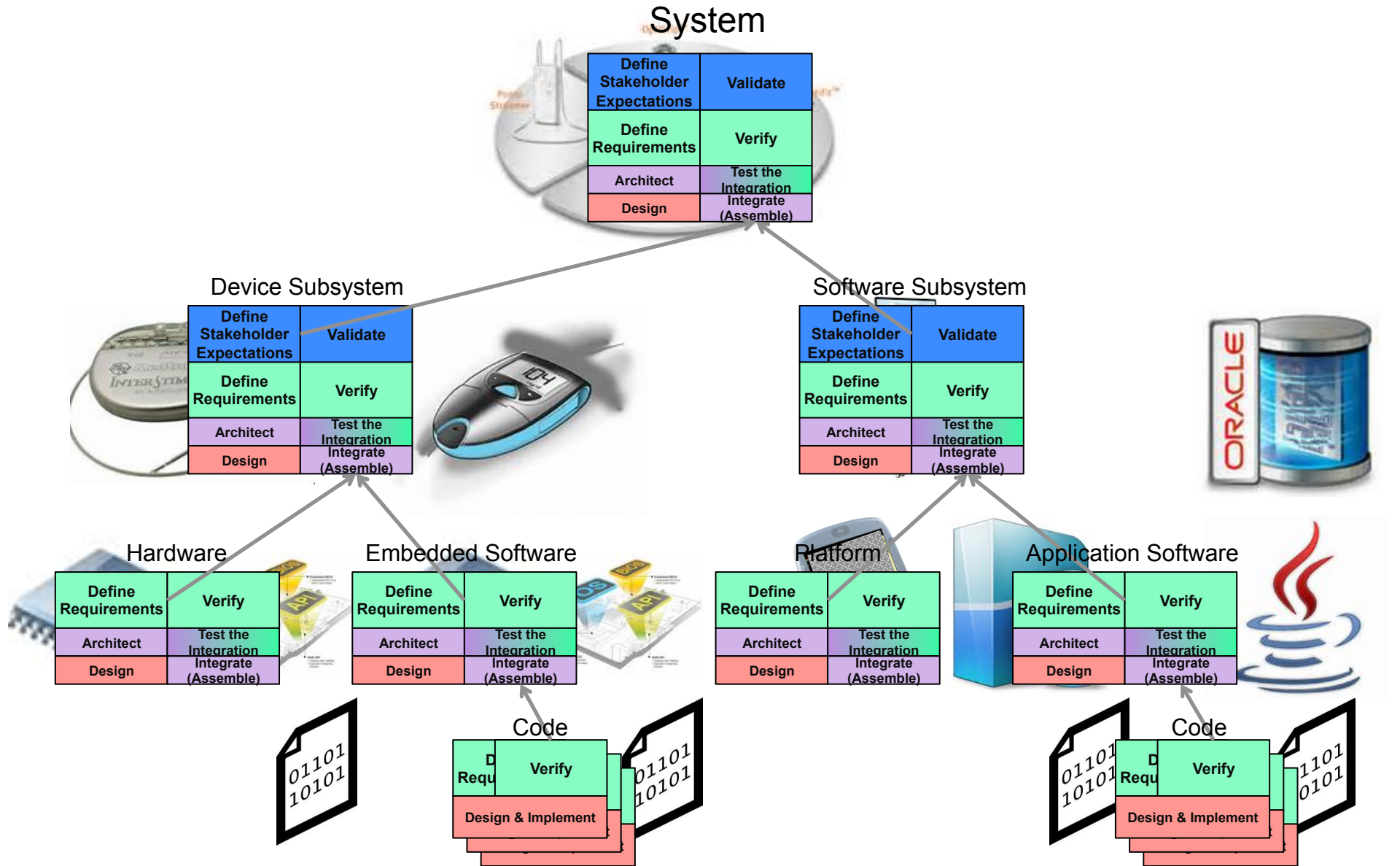

System
Define Stakeholder Expectations
Validate
Define Requirements
Verify
Architect
Integrate
Design
Implement
Subsystem
Define Stakeholder Expectations
Validate
Define Requirements
Verify
Architect
Integrate
Design
Implement
Subsystem
Define Stakeholder Expectations
Validate
Define Requirements
Verify
Architect
Integrate
Design
Implement
Subsystems We Integrate With But Don't Build
Define Requirements
Architect
Design


System
Define Stakeholder Expectations
Validate
Define Requirements
Verify
Architect
Integrate
Design
Implement
Subsystem
Define Stakeholder Expectations
Validate
Define Requirements
Verify
Architect
Integrate
Design
Implement
Subsystem
Define Stakeholder Expectations
Validate
Define Requirements
Verify
Architect
Integrate
Design
Implement
Components We Build
Verify
Implement
Subsystem
Define Stakeholder Expectations
Validate
Define Requirements
Verify
Architect
Integrate
Design
Implement
Components We Buy / Re-Use
Define Requirements
Design
Subsystems We Integrate With But Don't Build
Define Requirements
Architect
Design

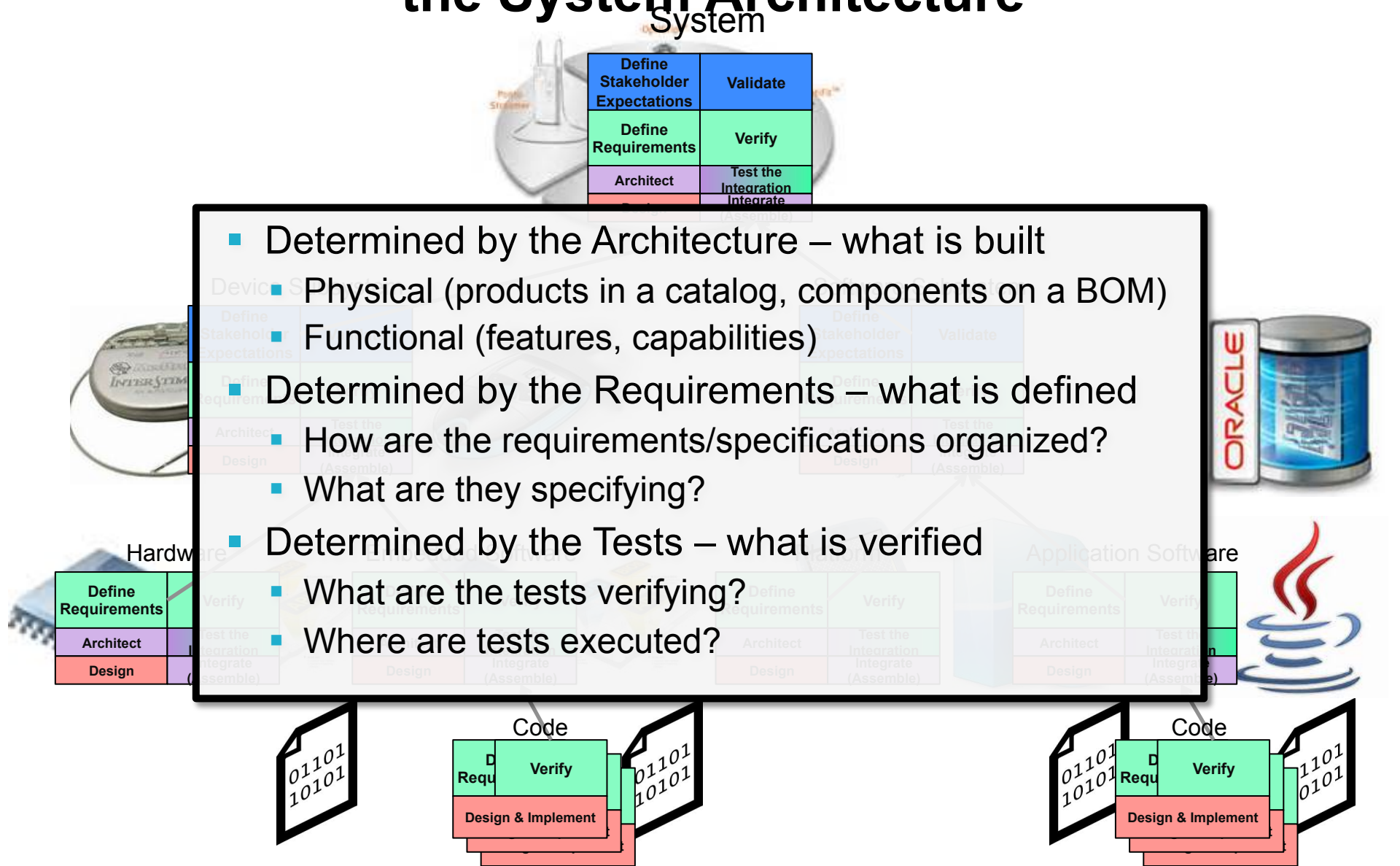
What is the System?



What is the System?
And What are the Activities & Deliverables?



The System Development Activities and Deliverables are Determined by the System Architecture



- Determined by the Architecture – what is built
  - Physical (products in a catalog, components on a BOM)
  - Functional (features, capabilities)
- Determined by the Requirements – what is defined
  - How are the requirements/specifications organized?
  - What are they specifying?
- Determined by the Tests – what is verified
  - What are the tests verifying?
  - Where are tests executed?

What is Your System?

System

Define Stakeholder Expectations	Validate
Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

Device Subsystem

Define Stakeholder Expectations	Validate
Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

Software Subsystem

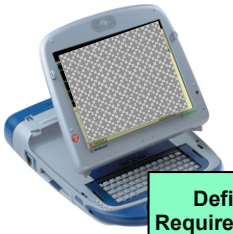
Define Stakeholder Expectations	Validate
Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

Hardware

Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

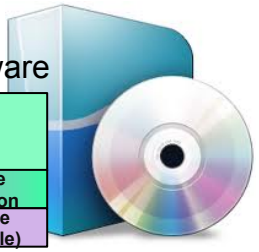
Embedded Software

Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)



Platform

Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)



Application Software

Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

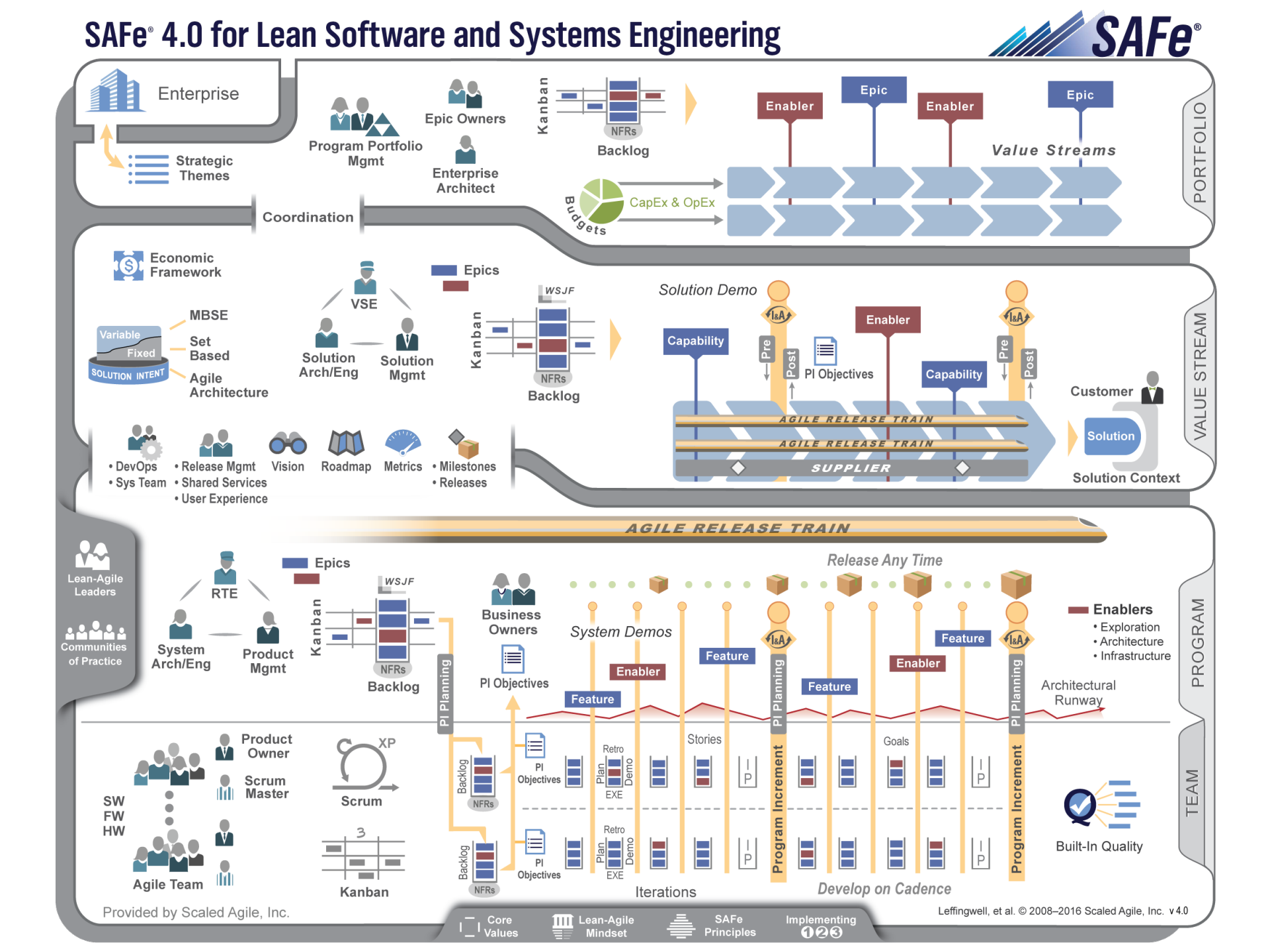
Code

Define Requirements	Verify
Design & Implement	

Code

Define Requirements	Verify
Design & Implement	

SAFe® 4.0 for Lean Software and Systems Engineering



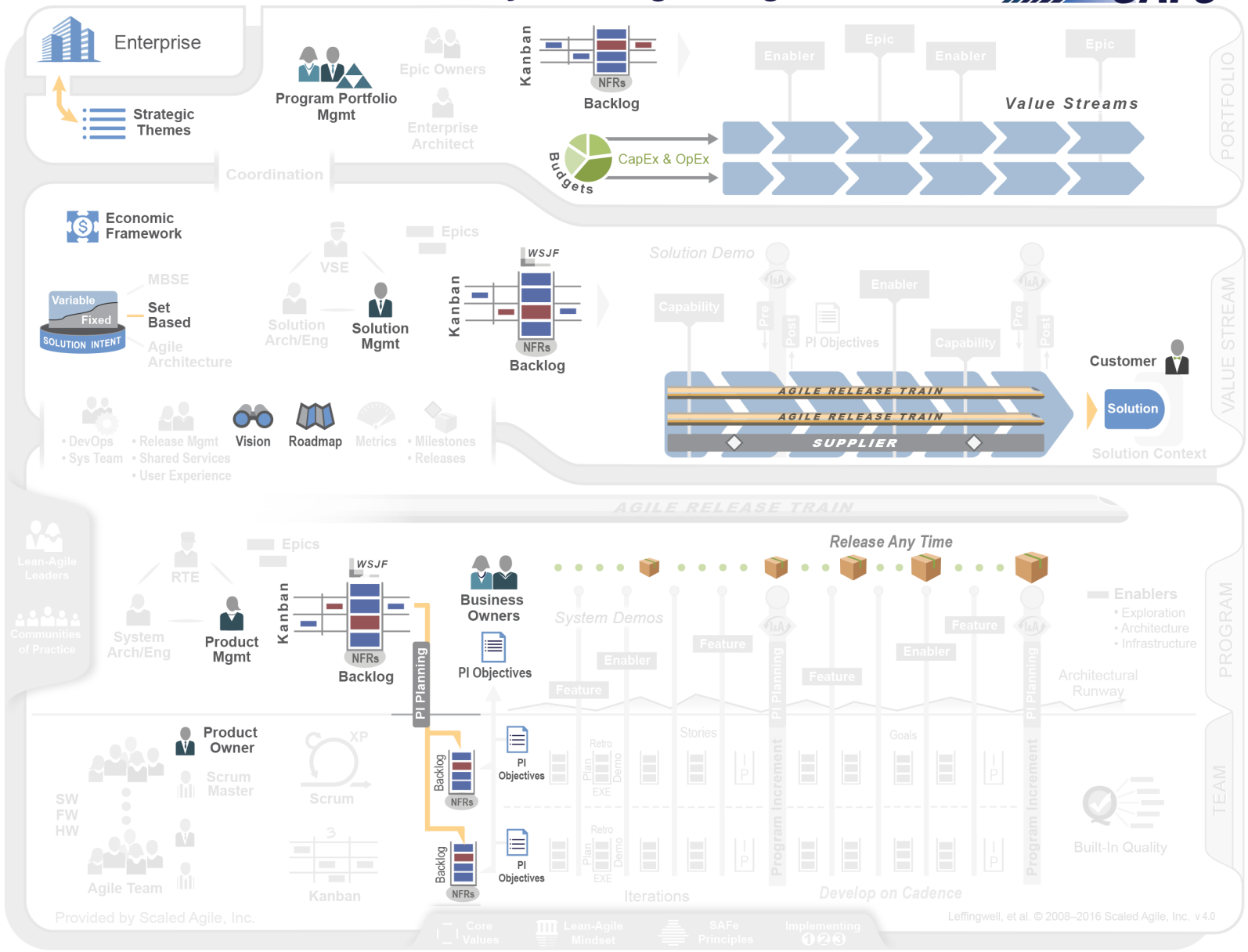
Overview of SAFe 4.0

- “SAFe 4.0 in 8 Pictures”
  - http://scaledagileframework.com/videos-and-presentations/

Applying SAFe to the Development of a System

- What is the System?
- What is the Value Stream(s)?
- What is the ART(s)?

SAFe® 4.0 for Lean Software and Systems Engineering



What is Your System?

System

Define Stakeholder Expectations	Validate
Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

Device Subsystem

Define Stakeholder Expectations	Validate
Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

Software Subsystem

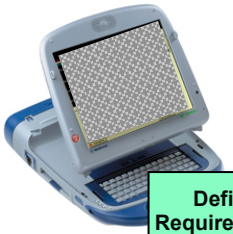
Define Stakeholder Expectations	Validate
Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

Hardware

Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

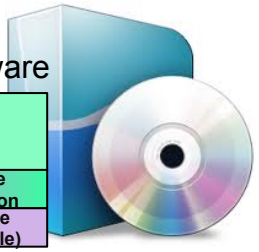
Embedded Software

Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)



Platform

Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)



Application Software

Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

Code

Define Requirements	Verify
Design & Implement	

Code

Define Requirements	Verify
Design & Implement	

What is Your System?

System

Define Stakeholder Expectations	Validate
Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

Value Stream
One ART?

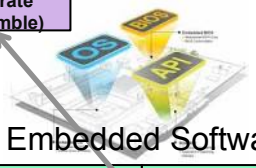
Subsystem = ART?

Device Subsystem

Define Stakeholder Expectations	Validate
Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

Hardware

Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

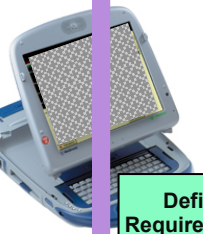


Embedded Software

Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

Code

Define Requ	Verify
Design & Implement	



Subsystem = ART?

Software Subsystem

Define Stakeholder Expectations	Validate
Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

Platform

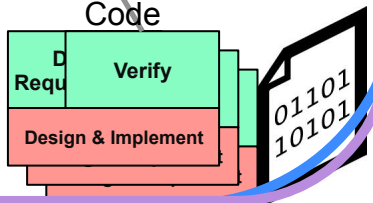
Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

Application Software

Define Requirements	Verify
Architect	Test the Integration
Design	Integrate (Assemble)

Code

Define Requ	Verify
Design & Implement	

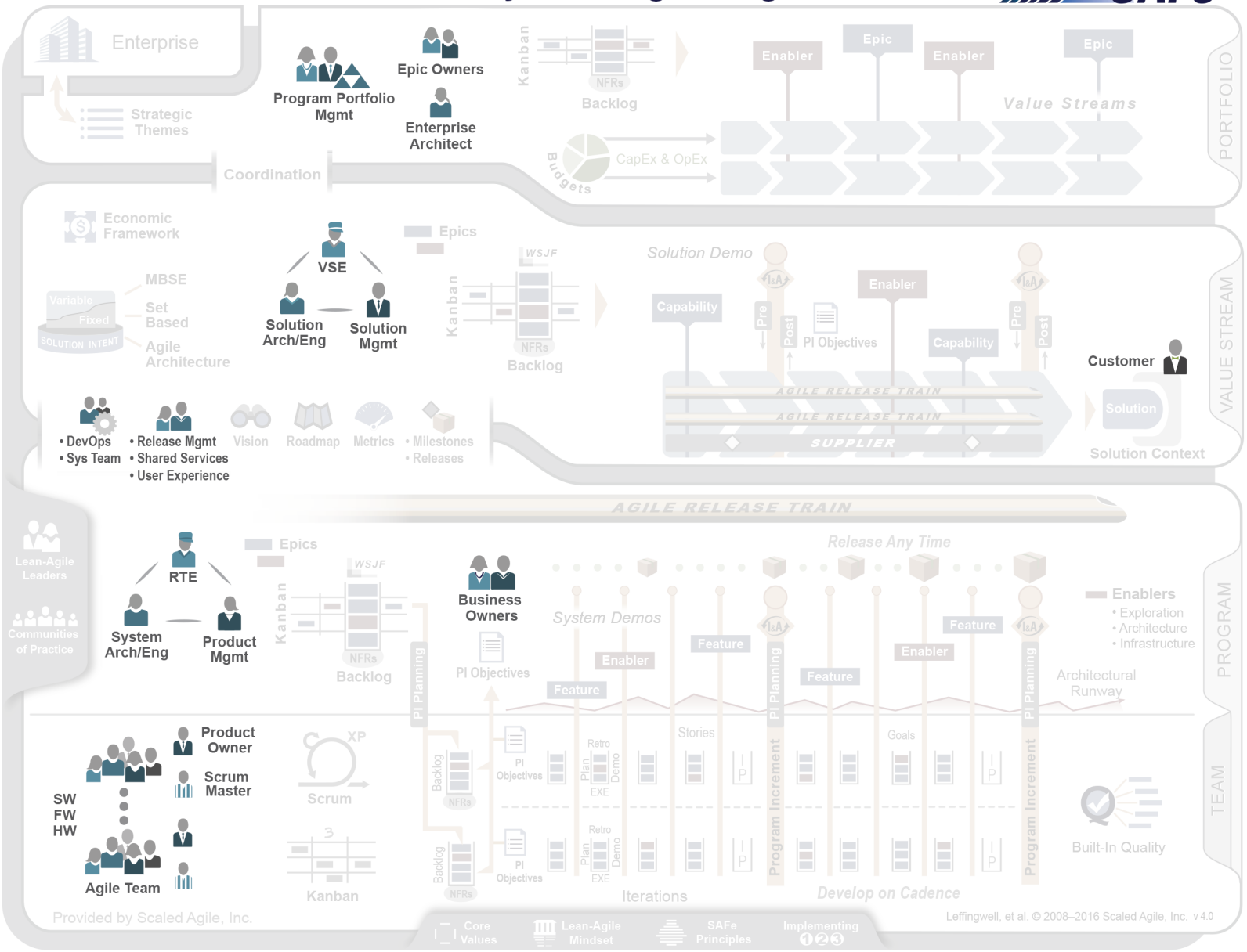


Applying SAFe to the Development of a System

- What are the Teams?
  - What do they deliver?
  - What activities do they perform?

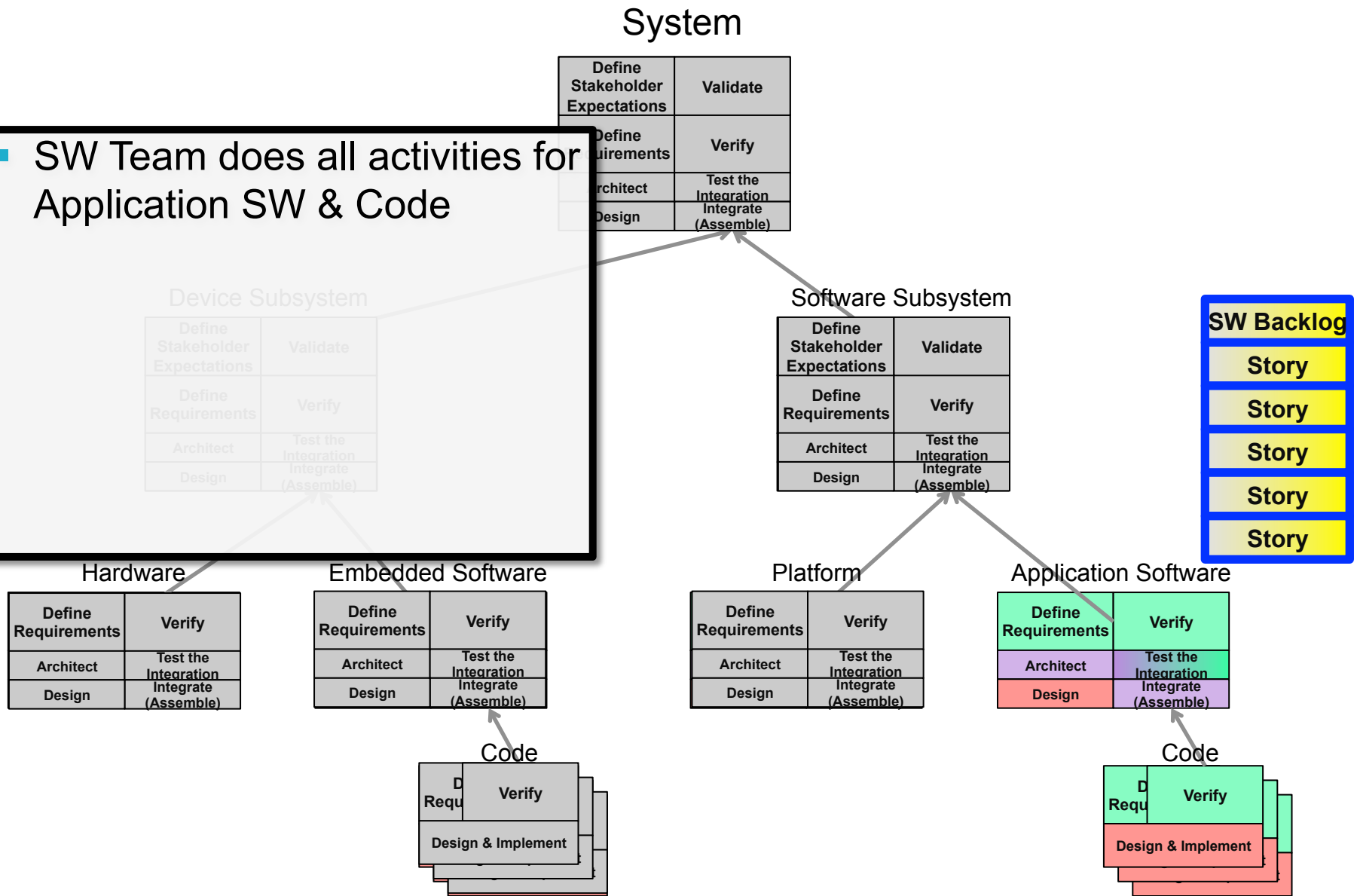
- What activities provide input to the teams?
  - And who does them?

SAFe® 4.0 for Lean Software and Systems Engineering



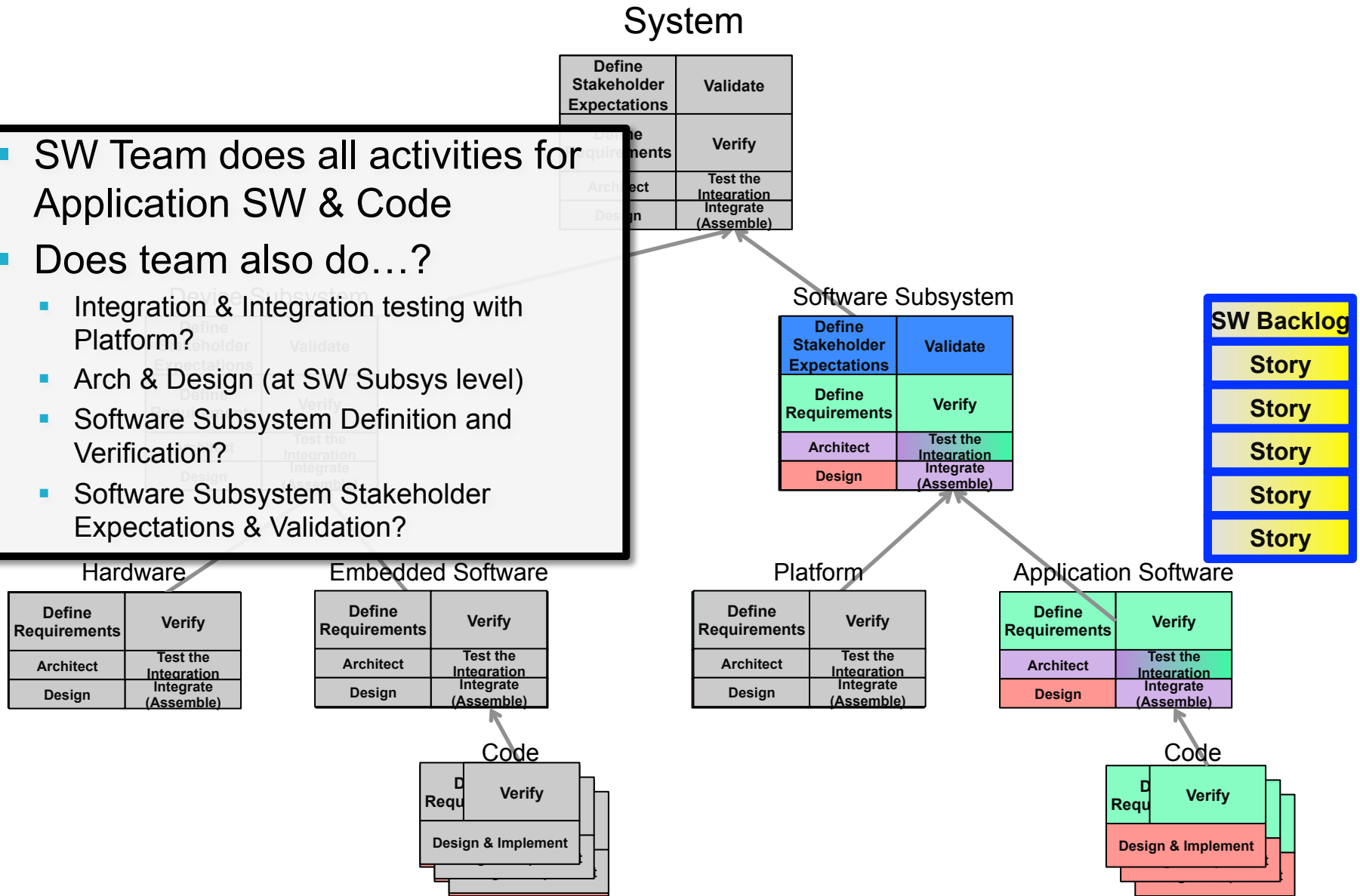
Application Software Team

- SW Team does all activities for Application SW & Code



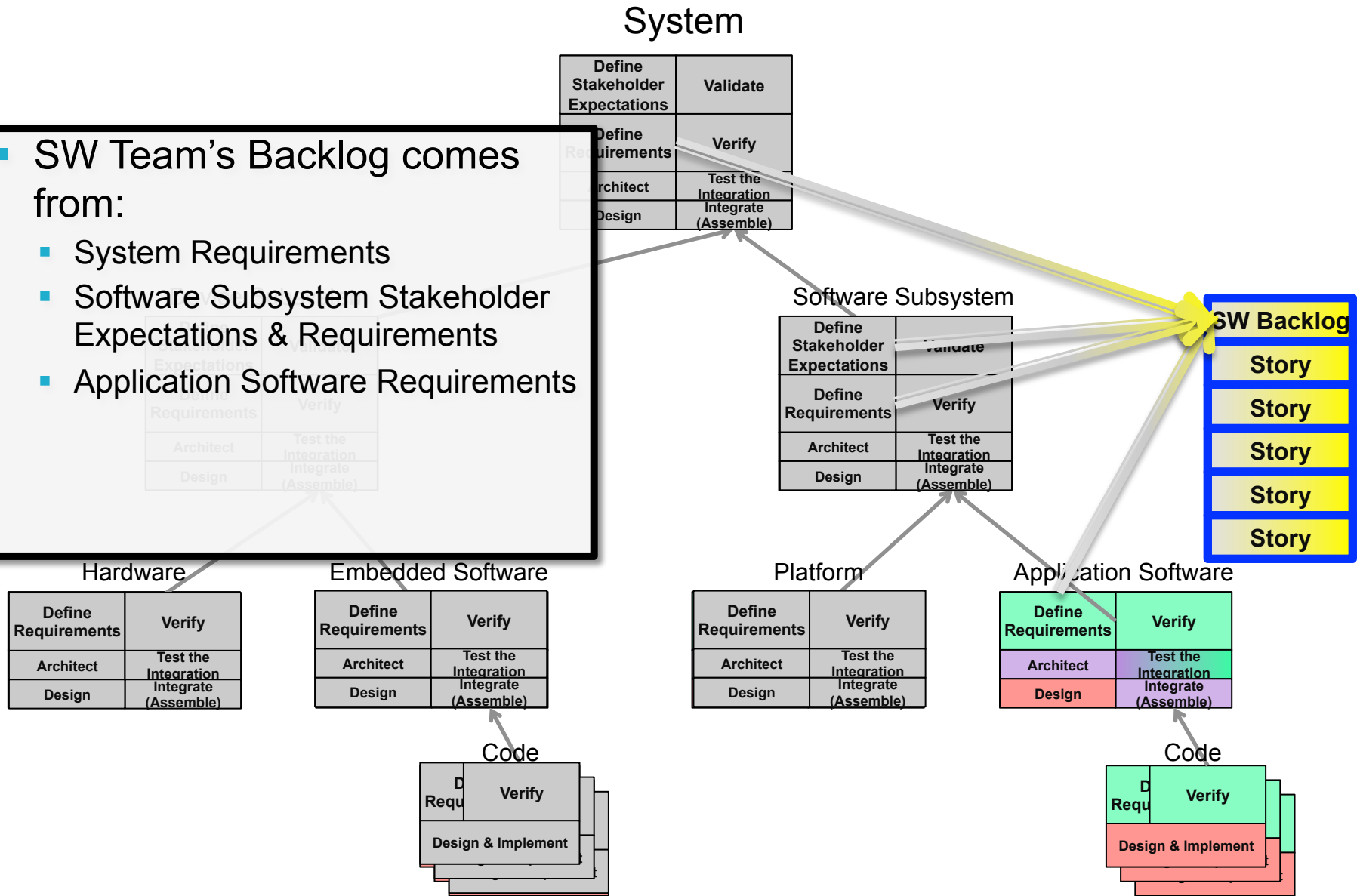
Application Software Team

- SW Team does all activities for Application SW & Code
- Does team also do…?
  - Integration & Integration testing with Platform?
  - Arch & Design (at SW Subsys level)
  - Software Subsystem Definition and Verification?
  - Software Subsystem Stakeholder Expectations & Validation?



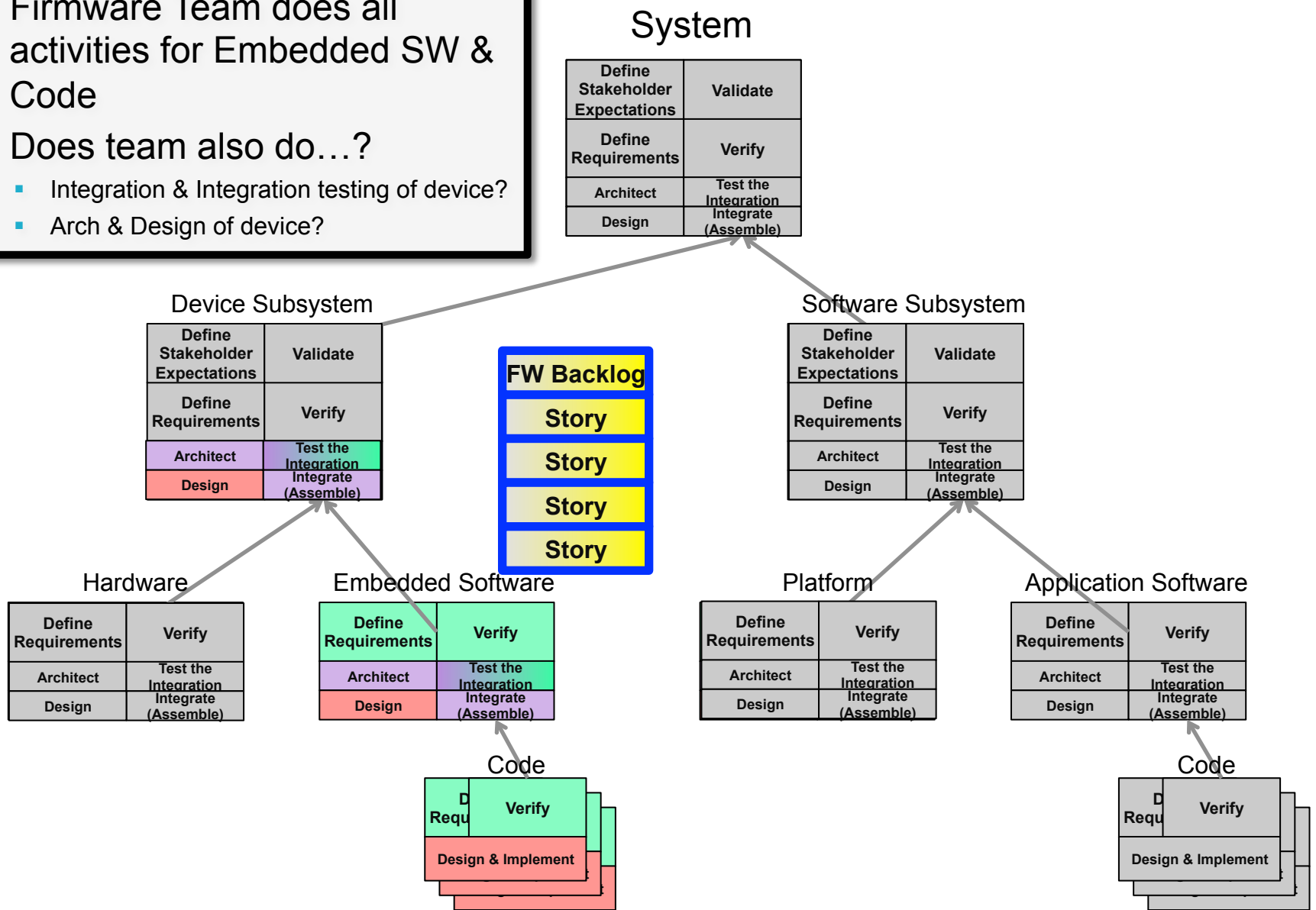
Application Software Team

- SW Team's Backlog comes from:
  - System Requirements
  - Software Subsystem Stakeholder Expectations & Requirements
  - Application Software Requirements



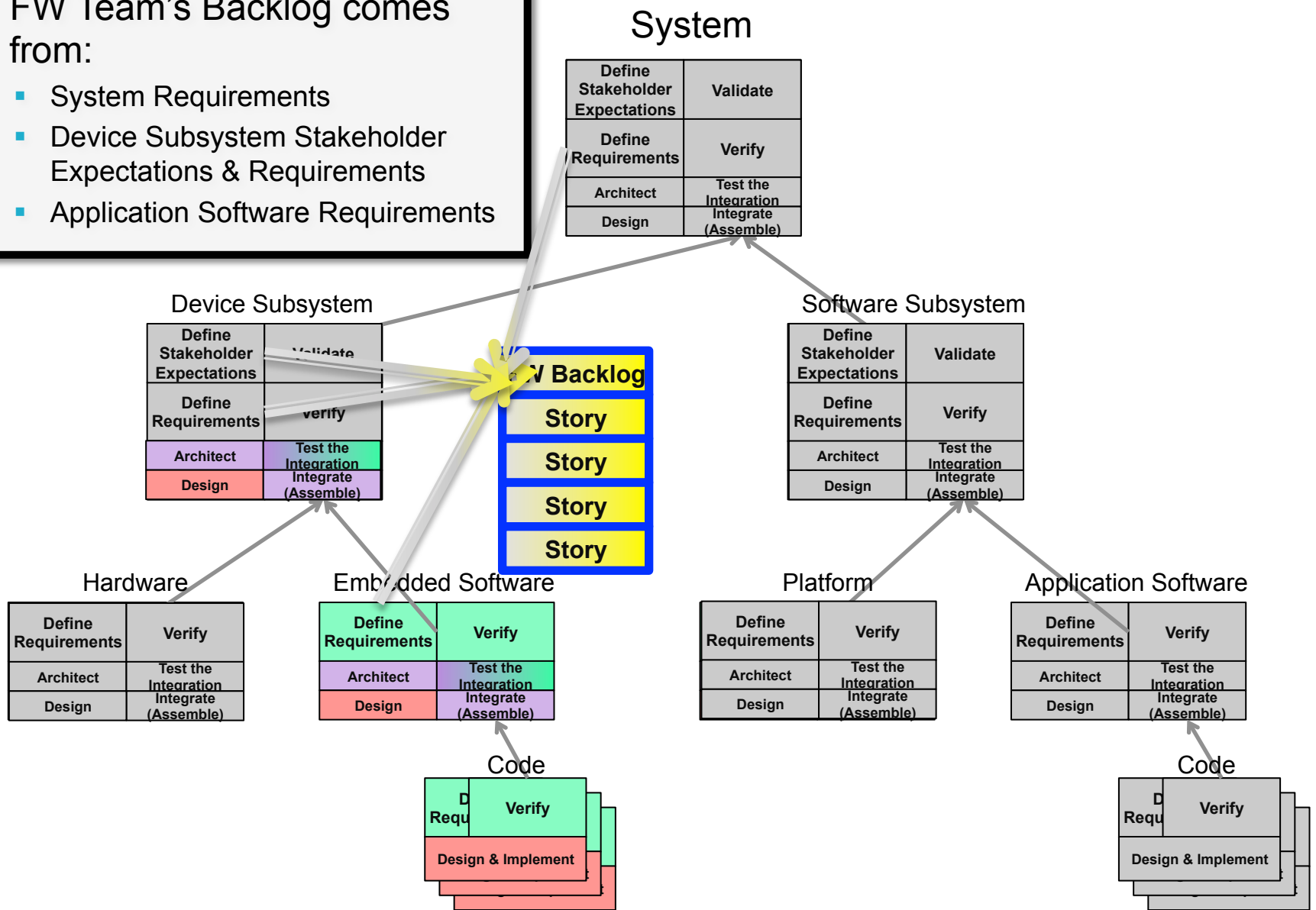
Embedded Software Team

- Firmware Team does all activities for Embedded SW & Code
- Does team also do…?
  - Integration & Integration testing of device?
  - Arch & Design of device?



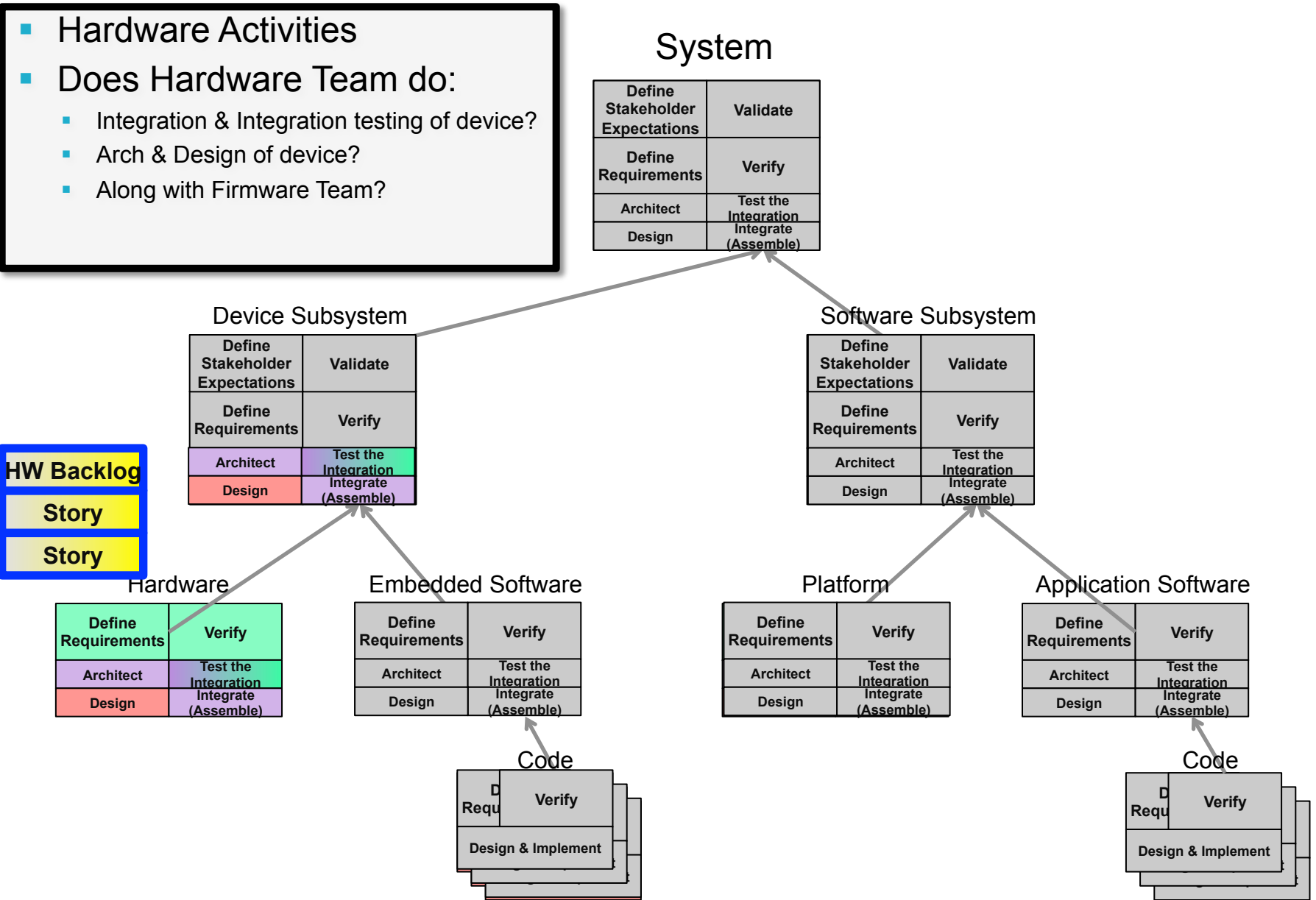
Embedded Software Team

- FW Team's Backlog comes from:
  - System Requirements
  - Device Subsystem Stakeholder Expectations & Requirements
  - Application Software Requirements

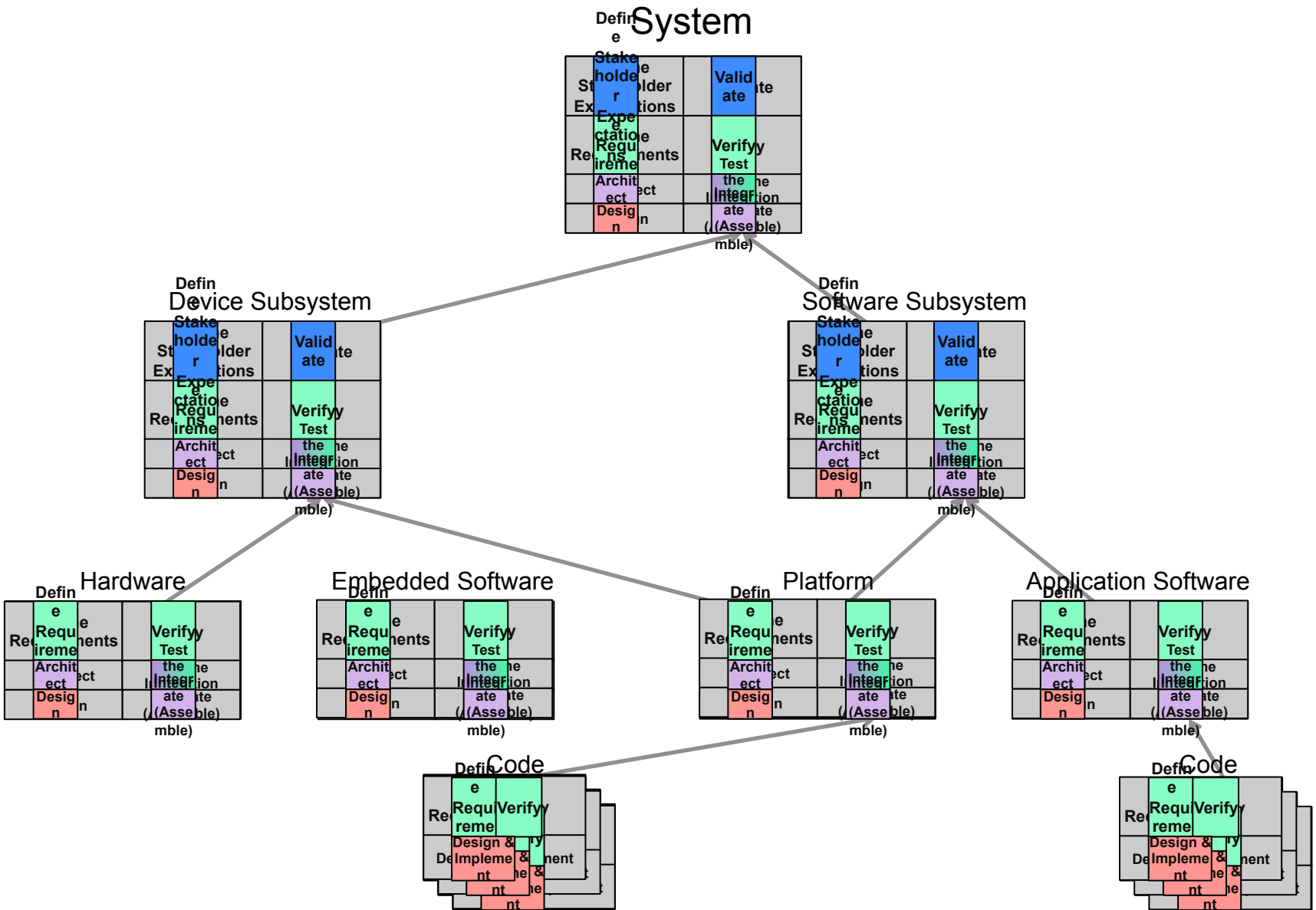


Hardware Team

- Hardware Activities
- Does Hardware Team do:
  - Integration & Integration testing of device?
  - Arch & Design of device?
  - Along with Firmware Team?



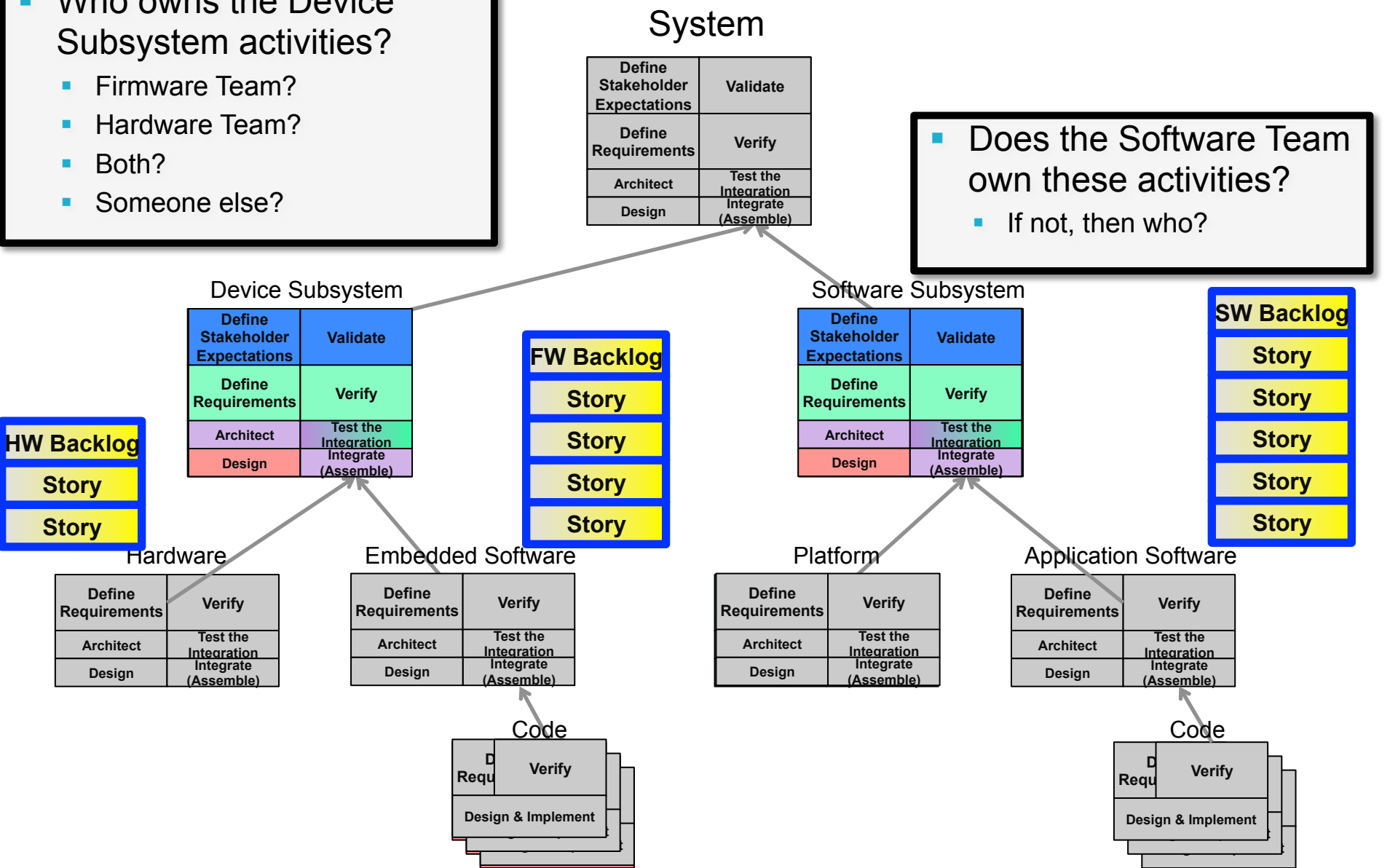
Cross-Functional Feature Team



Subsystem Activities – What team/role?

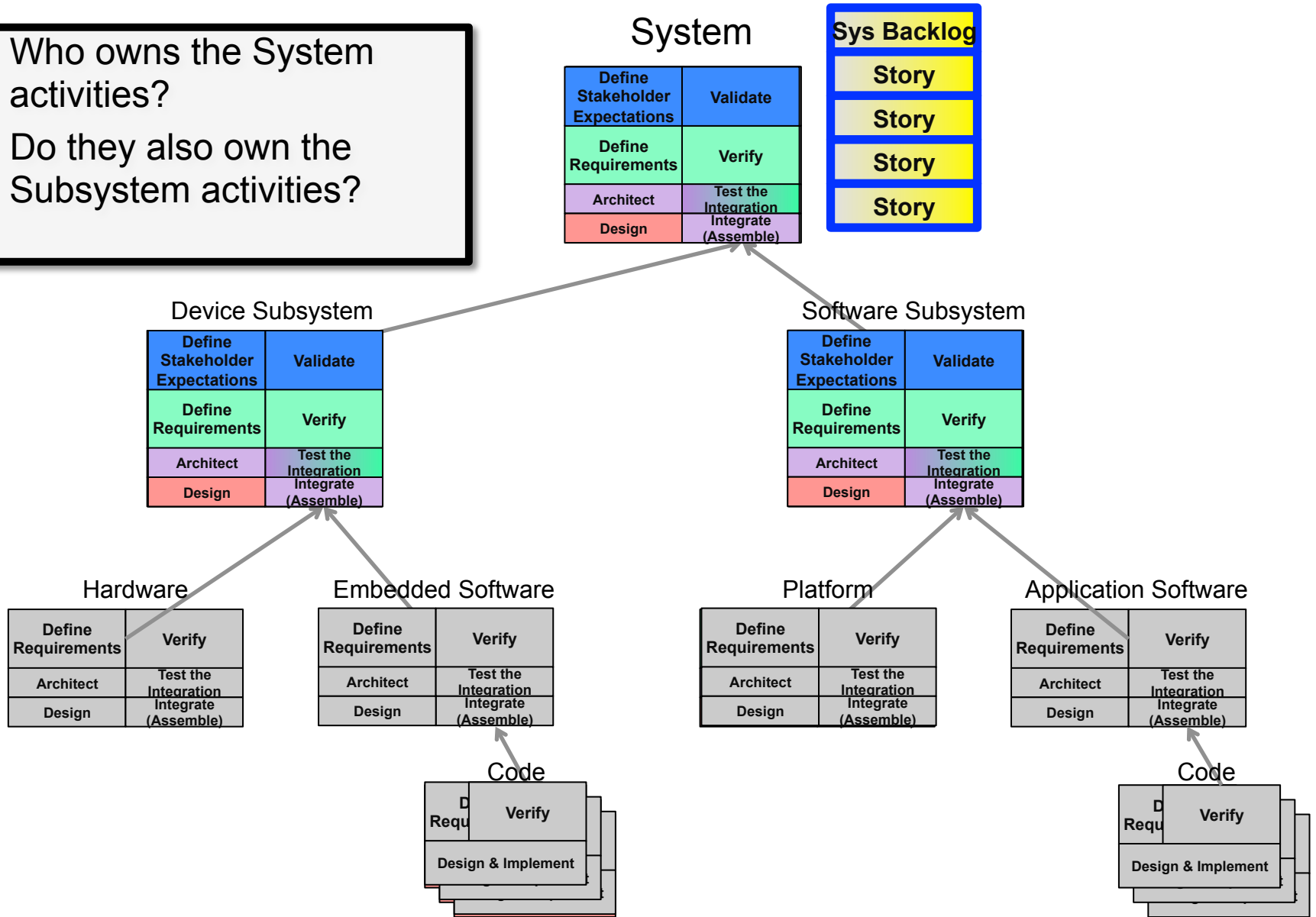
- Who owns the Device Subsystem activities?
  - Firmware Team?
  - Hardware Team?
  - Both?
  - Someone else?

- Does the Software Team own these activities?
  - If not, then who?

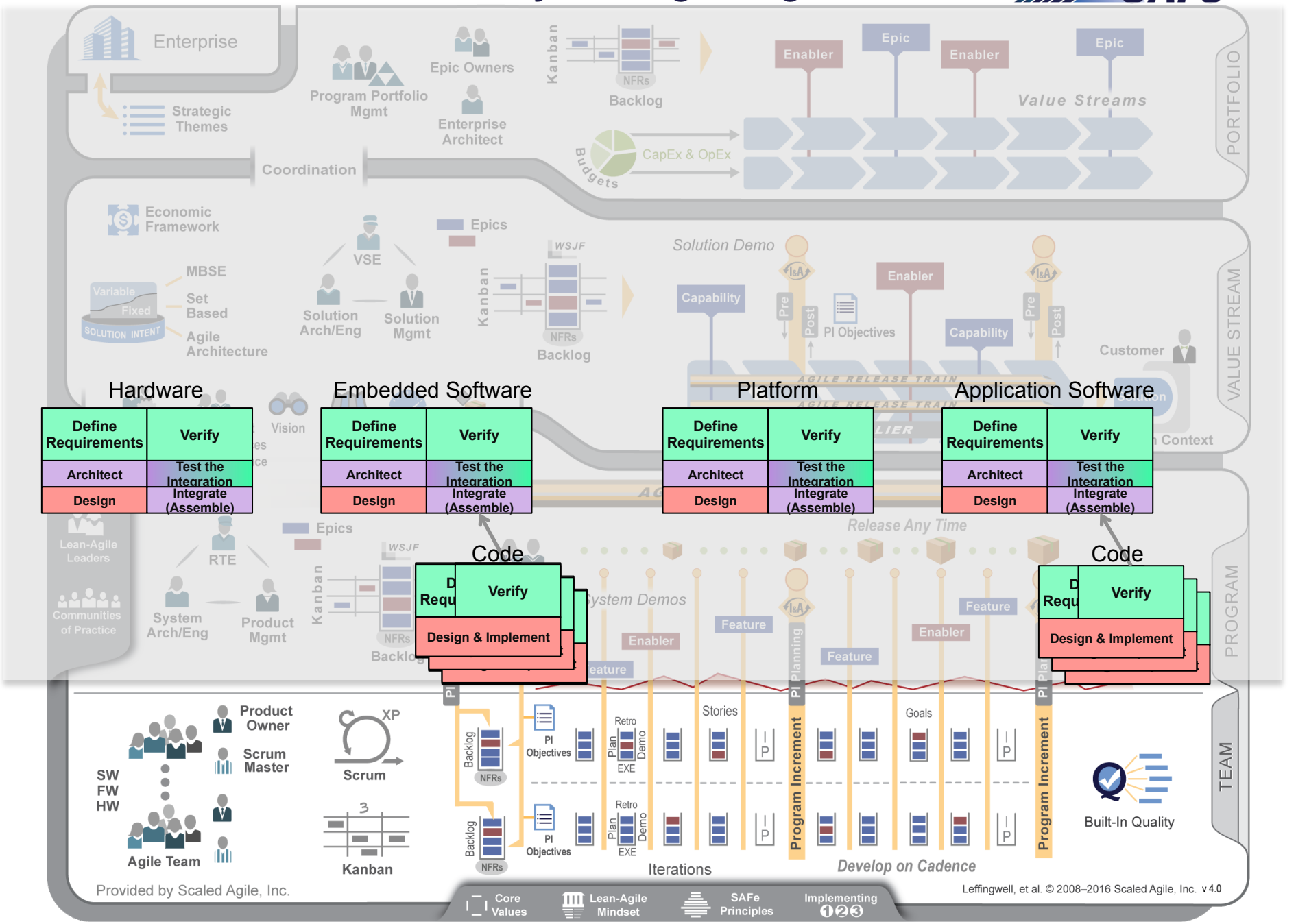


System Activities – What Team/Role

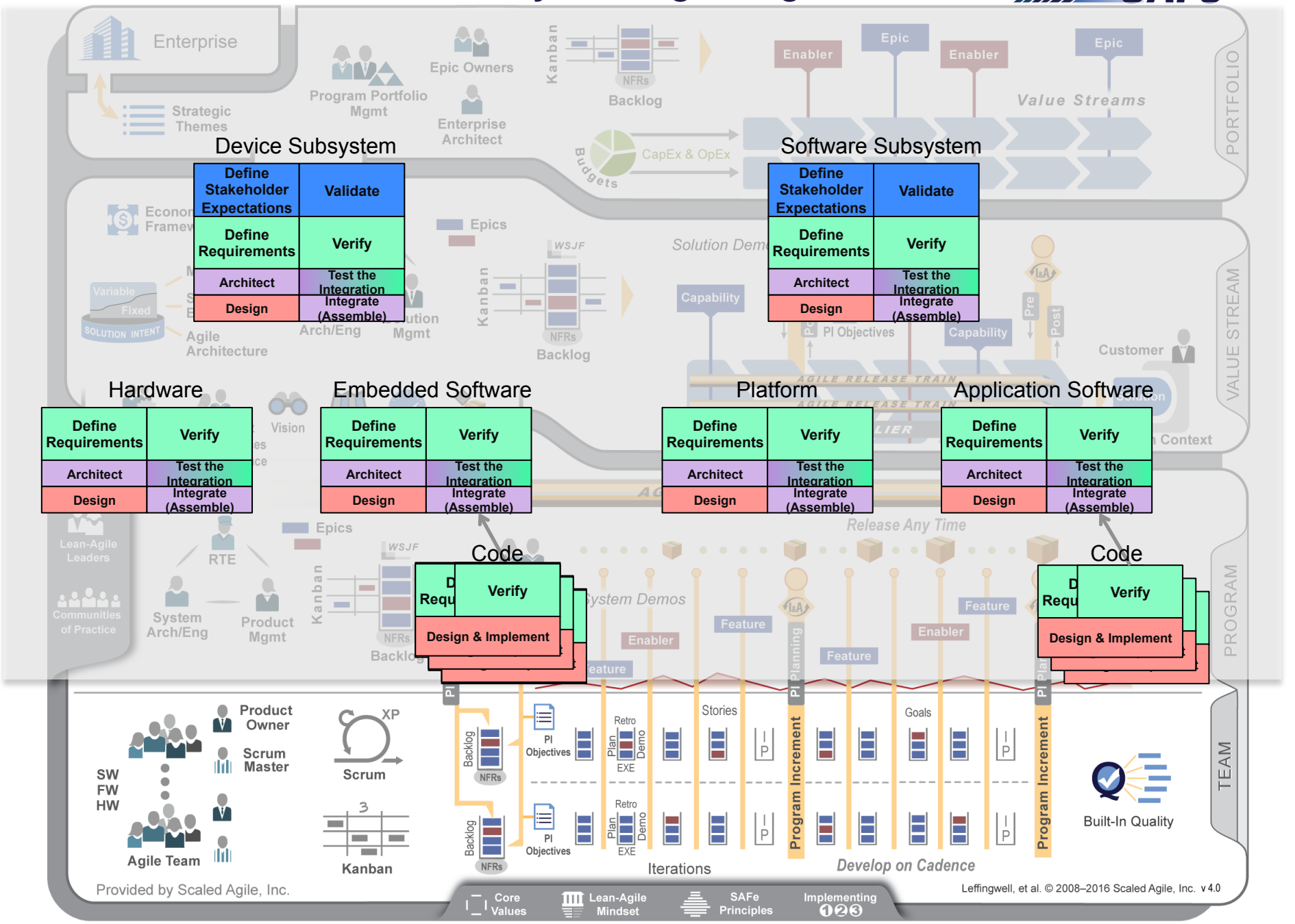
- Who owns the System activities?
- Do they also own the Subsystem activities?



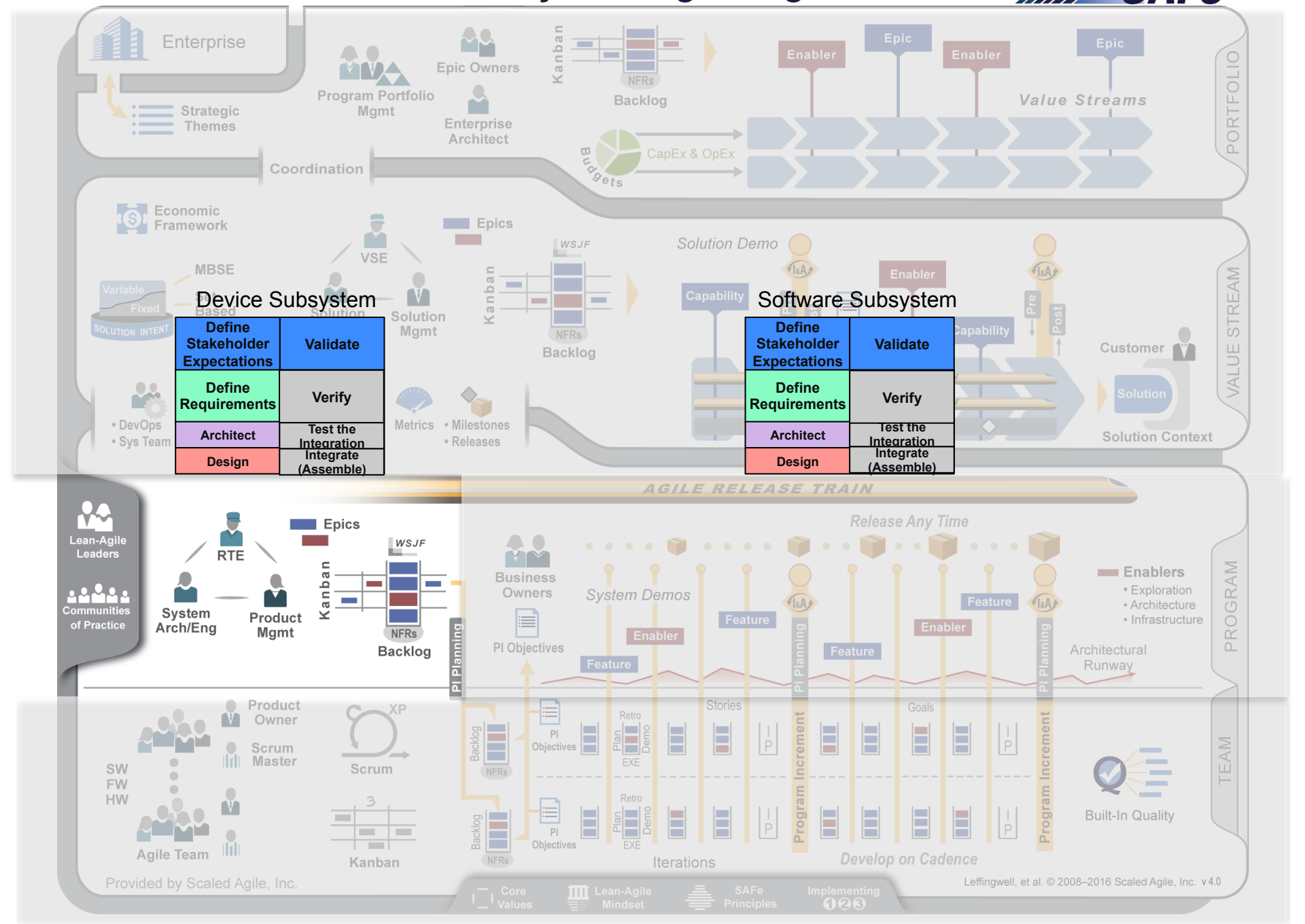
SAFe® 4.0 for Lean Software and Systems Engineering



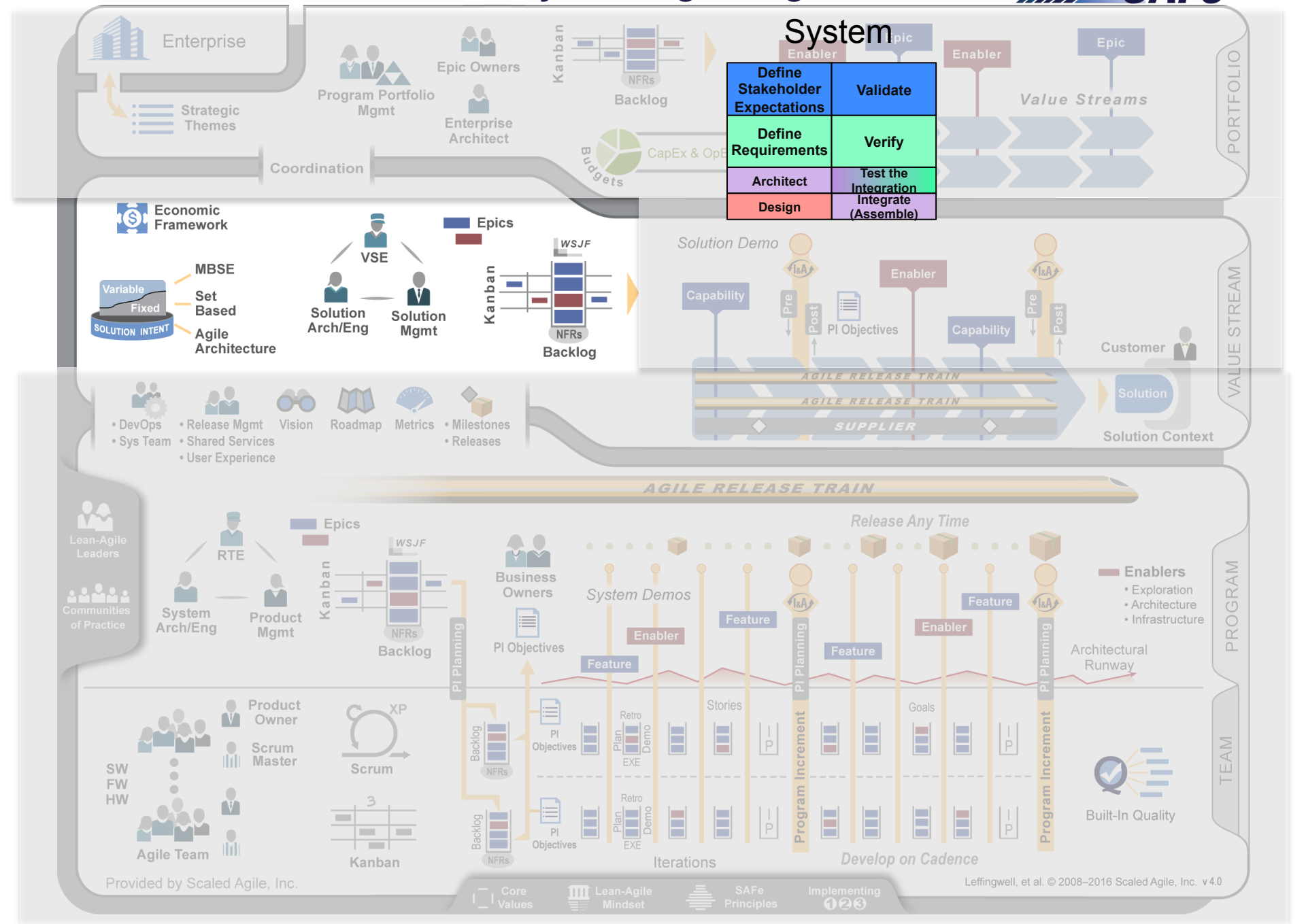
SAFe® 4.0 for Lean Software and Systems Engineering



SAFe® 4.0 for Lean Software and Systems Engineering



SAFe® 4.0 for Lean Software and Systems Engineering



Guidelines for Team Structure

- Team size
  - 3-9 (the Scrum rule) - “The 2 Pizza Rule”
- What is their Product?
  - What do they deliver? What do they demo?
- Who is their Product Owner? Stakeholders?
  - To whom do they deliver? Who is invited to the demo?
- What Activities must they perform? Are they capable?
  - Expertise & experience. Specialists and Generalists. Access to visitors/helpers when needed.
- Development Teams are a System
  - Define the boundaries, manage the interfaces

Agile for System Development

- Foundations, Good Ideas and Conversation Starters

- Did this enlighten you on some Foundational topics (SAFe, Systems Engineering Models)?
- Did this spark some Good Ideas?
- If you weren't here for the Conversation, well, these slides might not …

Contact Information

- Kelly Weyrauch
- Kelly@AgileQualitySystems.com
- 763-688-0980

- www.AgileQualitySystems.com

- Connect with me on LinkedIn