321 Gang | THE RATIONAL EXPERTS

Lean Systems Engineering Using the Scaled Agile Framework (SAFe)

Harry Koehnemann
Director of Technology
321Gang
harry@321gang.com

We Focus on the Wrong Things

- Assume variability is bad – schedule, requirements, etc.
  - Don’t see the value of change
- Assume efficiency is good, but blind to queues
  - Assume lost productivity is waste
- Assume reduce variability == reduce risk
  - Risk aversions restricts innovation, $$
  - Choose reduced schedule variability over profitability
- Solutions create waste
  - More management, reports, process

© LaughingStock International Inc./dist. by UFS Inc., 2010

“Are you steering?”

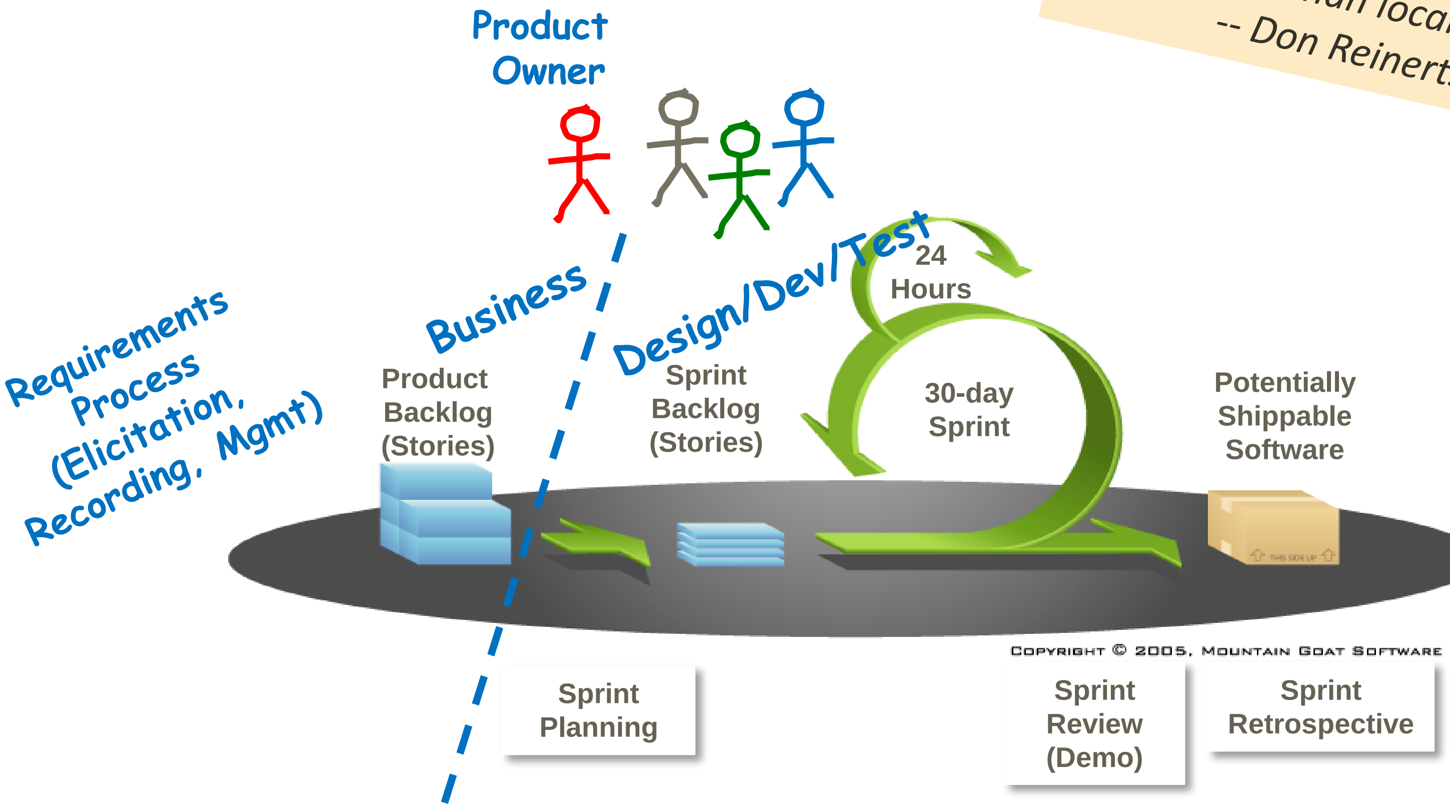
We Collaborate the Wrong Way

- System decides, creates specs, subsystems build
- Decide early
- Deliver slow and in large batches
- Obtain knowledge in stages; mitigate (real) risks at end
- Wasteful
  - Extra features, waiting, more management,
  - Many work handoff, usually in big chunks, detailed at wrong time
- Efficiency gains by “experts” optimizing each step
  - No team ownership of process
  - No “optimize the whole”

Where Does Agile Excel?

- Answer: Alignment and Collaboration
- But, only solves team-level alignment and collaboration

Principle of Alignment: There is more value created with overall alignment than local excellence
-- Don Reinertsen



Product Owner
Requirements Process (Elicitation, Recording, Mgmt)
Business
Design/Dev/Test
Product Backlog (Stories)
Sprint Backlog (Stories)
24 Hours
30-day Sprint
Potentially Shippable Software
Copyright © 2005, Mountain Goat Software
Sprint Planning
Sprint Review (Demo)
Sprint Retrospective

SAFe LSE Principles

Take an economic view

Apply systems thinking

Assume variability; preserve options

Develop systems incrementally; integrate and test frequently

Manage risk and efficacy with fast, synchronous learning cycles

Visualize and limit WIP, reduce batch sizes, and manage queue lengths

Base milestones on objective evaluation of working systems

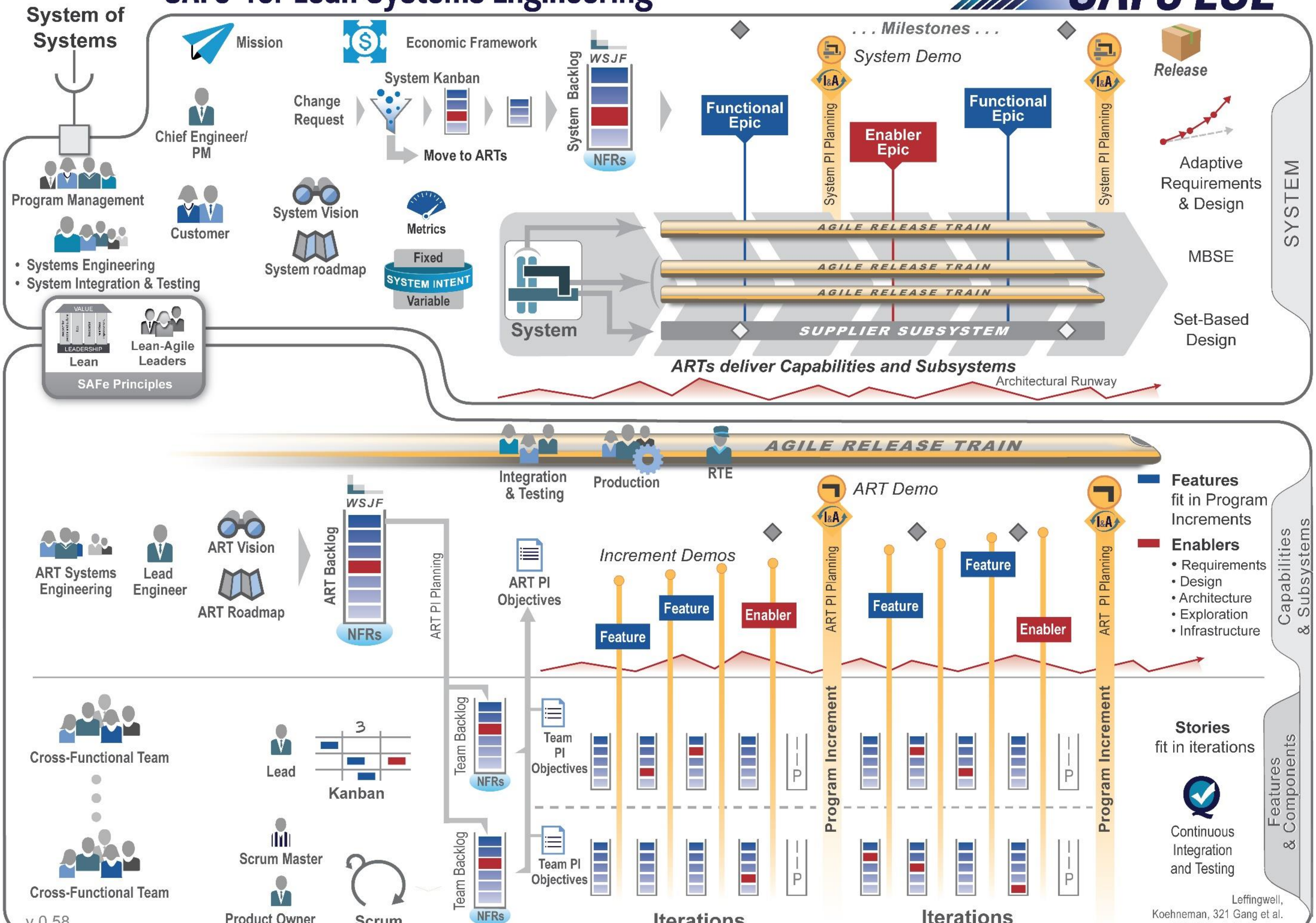
Synchronize with cross-domain planning and collaboration

Unlock the intrinsic motivation of knowledge workers

Decentralize decision-making

SAFe® for Lean Systems Engineering

SAFe® LSE



System of Systems

Program Management

- Systems Engineering
- System Integration & Testing

Chief Engineer/ PM

Customer

Mission

Economic Framework

Change Request

System Kanban

Move to ARTs

System Vision

System roadmap

Metrics

Fixed

SYSTEM INTENT

Variable

WSJF

System Backlog

NFRs

. . . Milestones . . .

System Demo

I&A

System PI Planning

Functional Epic

Enabler Epic

Functional Epic

Release

Adaptive Requirements & Design

MBSE

Set-Based Design

SYSTEM

System

AGILE RELEASE TRAIN

AGILE RELEASE TRAIN

AGILE RELEASE TRAIN

SUPPLIER SUBSYSTEM

ARTs deliver Capabilities and Subsystems

Architectural Runway

Lean

Lean-Agile Leaders

SAFe Principles

AGILE RELEASE TRAIN

Integration & Testing

Production

RTE

ART Demo

Features fit in Program Increments

Enablers

- Requirements
- Design
- Architecture
- Exploration
- Infrastructure

Capabilities & Subsystems

ART Systems Engineering

Lead Engineer

ART Vision

ART Roadmap

WSJF

ART Backlog

NFRs

ART PI Planning

ART PI Objectives

Increment Demos

Feature

Feature

Enabler

Feature

Feature

Enabler

Program Increment

Program Increment

Cross-Functional Team

Cross-Functional Team

Lead

Kanban

3

Scrum Master

Product Owner

Scrum

Team Backlog

NFRs

Team Backlog

NFRs

Team PI Objectives

Team PI Objectives

IP

Iterations

Iterations

Stories fit in iterations

Continuous Integration and Testing

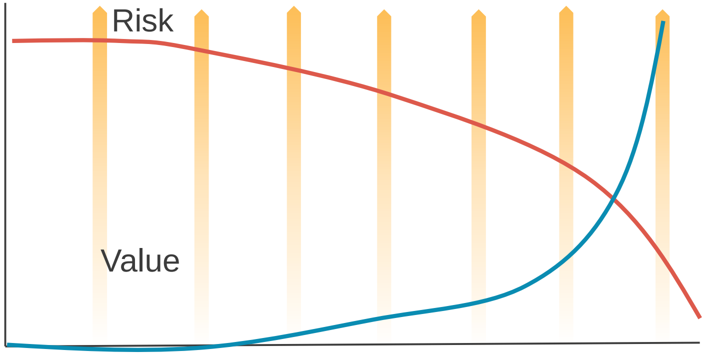
Features & Components

v 0.58

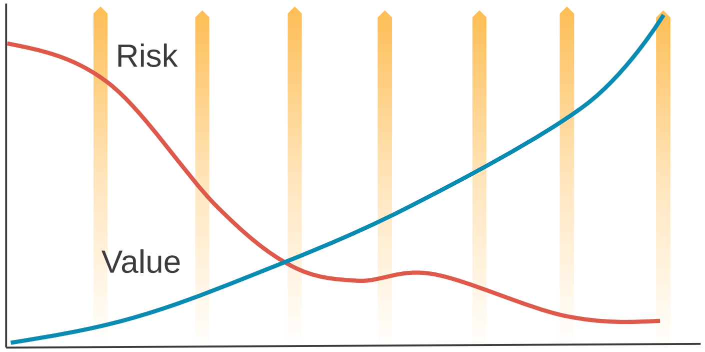
Leffingwell, Koehneman, 321 Gang et al.

Program Increments are Learning Cycles

- “Pull” event for entire system
  - Build what we understand
  - Build knowledge for what we don’t understand, and make decisions
  - Require both for feedback



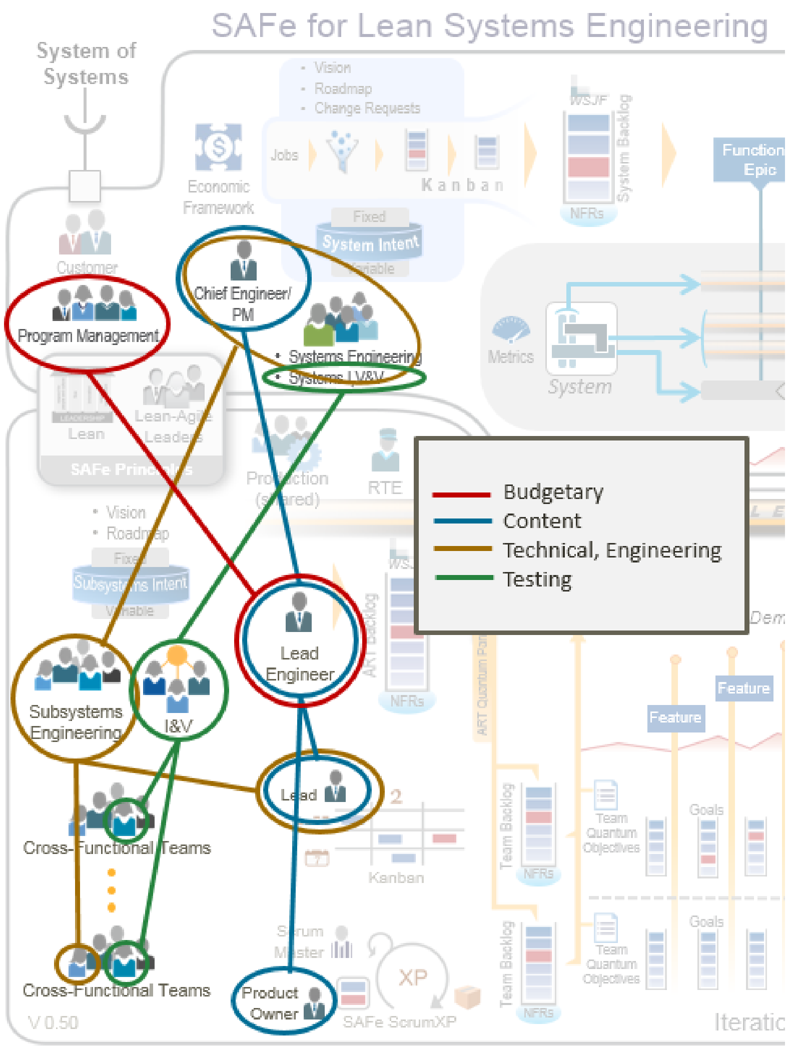
Waterfall



Lean-Agile

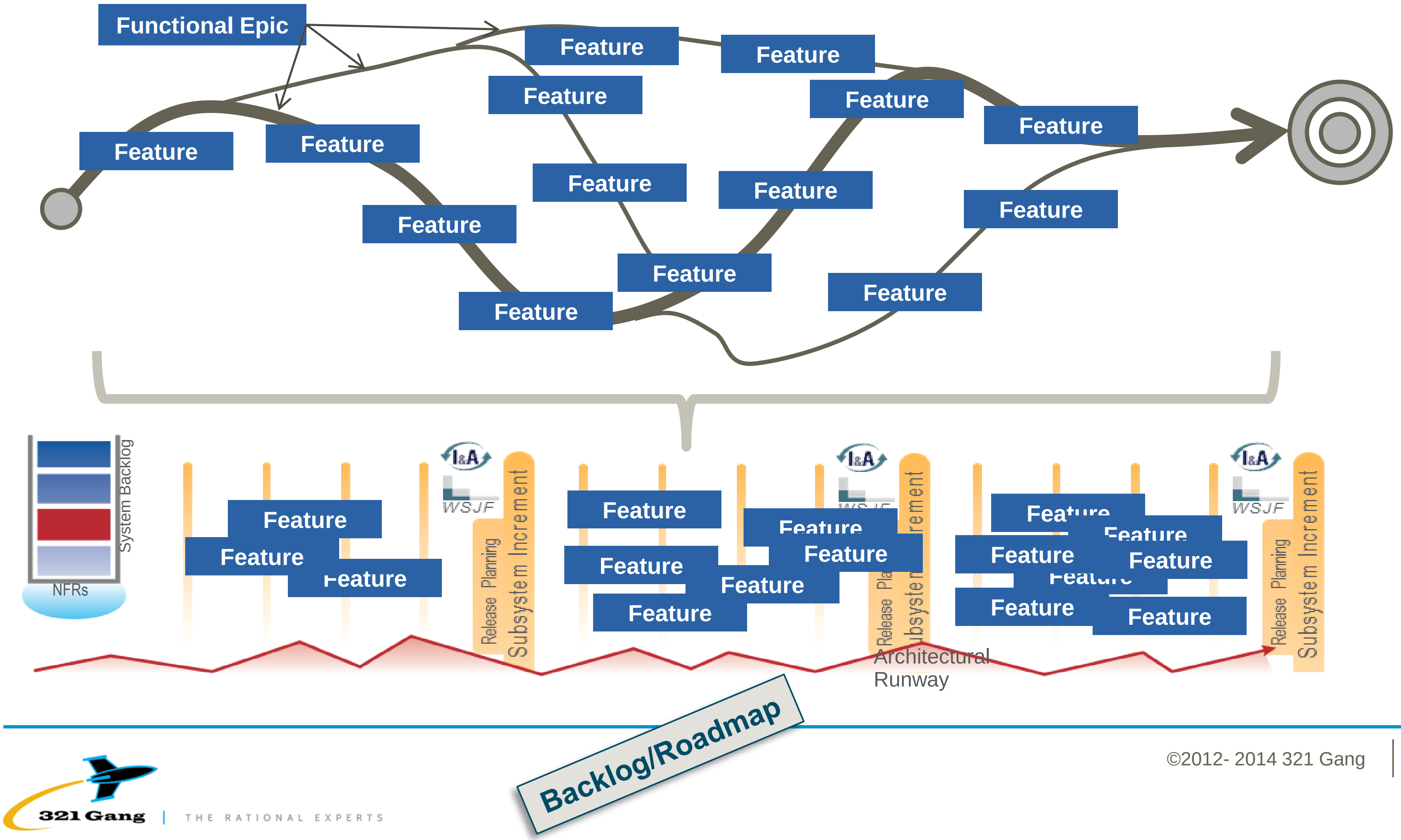
LSE Role Alignment

- Authority for budget, content, technical, and testing exist at each level
- These roles collaborate frequently for alignment



Epics with the Roadmap and Backlog

- Epics are functional threads/operational scenario for system
- Decompose into Features on Roadmap (schedule)



Functional Epic

Feature

Feature

Feature

Feature

Feature

Feature

Feature

Feature

Feature

Feature

Feature

Feature

Feature

Feature

System Backlog

NFRs

Feature

Feature

Feature

I&A

WSJF

Release Planning

Subsystem Increment

Feature

Feature

Feature

Feature

Feature

Feature

I&A

WSJF

Release Planning

Subsystem Increment

Feature

Feature

Feature

Feature

Feature

Feature

Feature

I&A

WSJF

Release Planning

Subsystem Increment

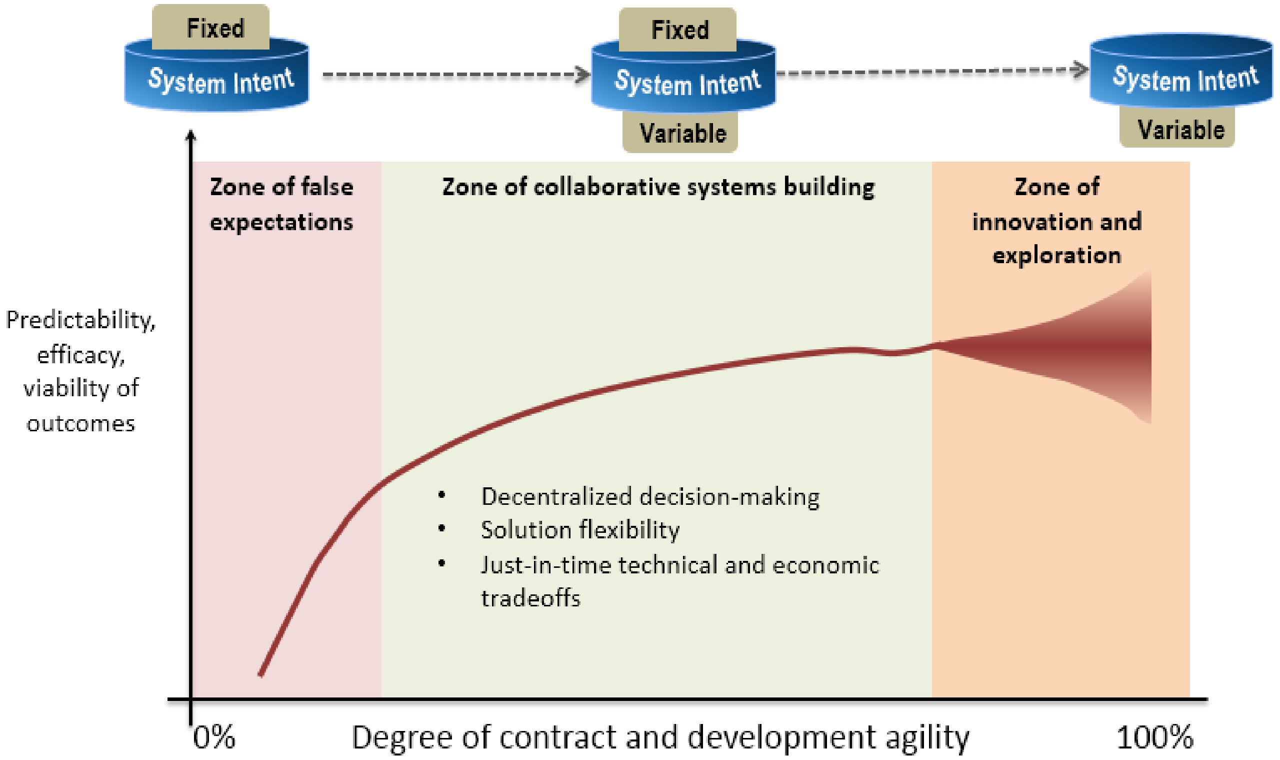
Architectural Runway

Backlog/Roadmap

321 Gang | THE RATIONAL EXPERTS

Adaptive Requirements and Design

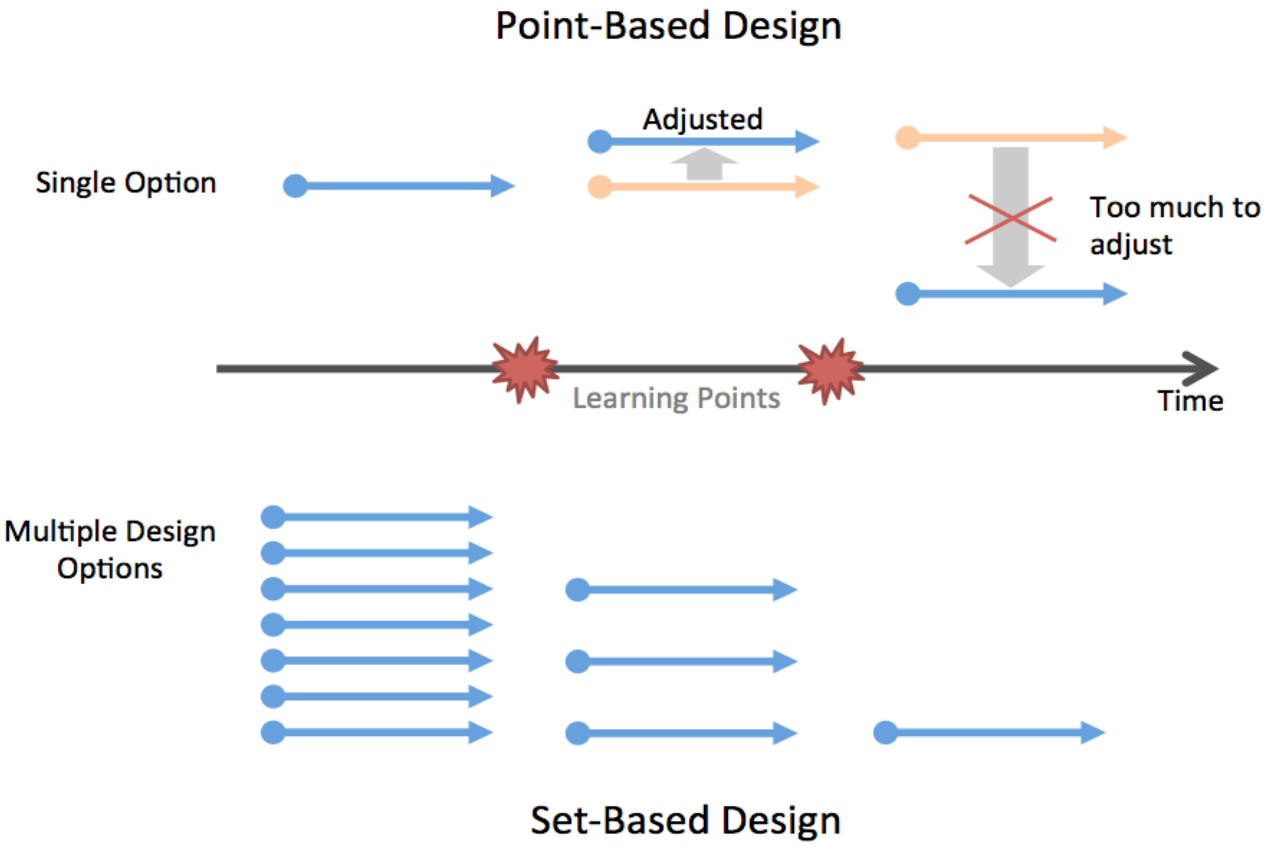
- Stage-gate processes creates *false expectations*
  - Complex system can be *fully* defined before they are built
  - We can build it right the *first* time, in fell swoop



- Decentralized decision-making
- Solution flexibility
- Just-in-time technical and economic tradeoffs

Set-based Design

- Maintains multiple design options through a longer period of development
- Better support fixed-schedule programs

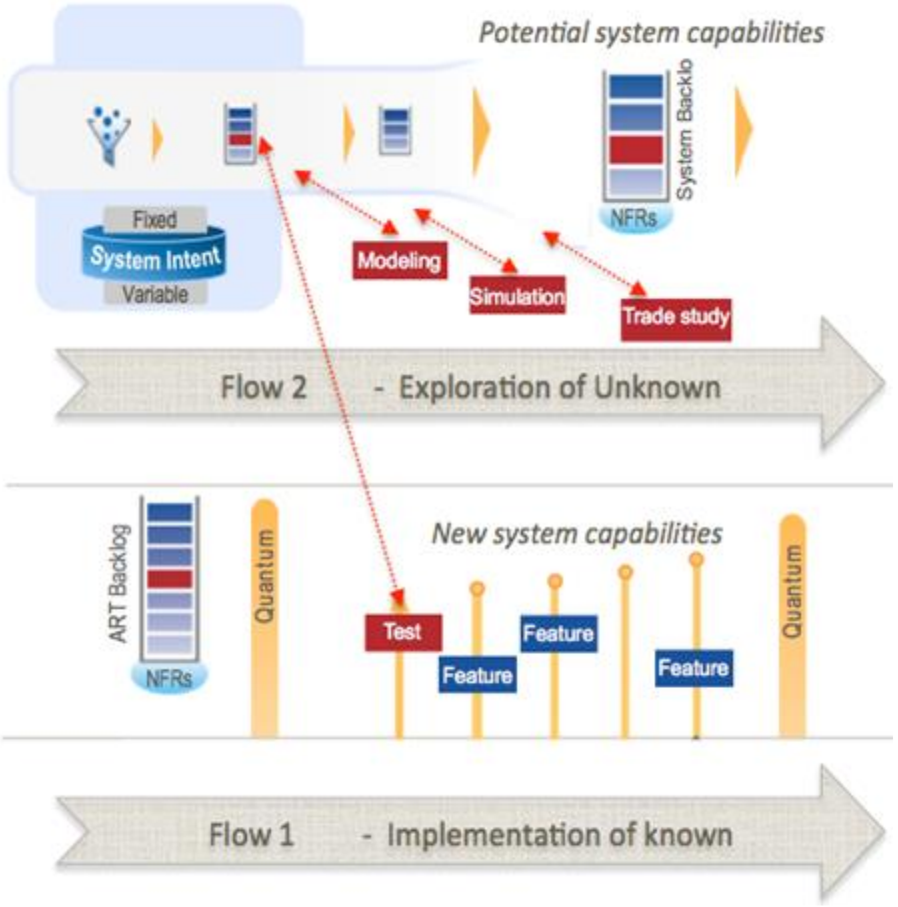


Moving From Fixed to Variable Req and Design

- Two parallel flows
  - Explore what we don't know
  - Implement what we do know

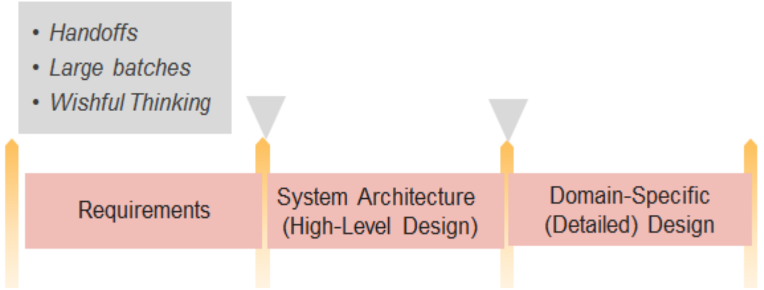
- Explore
  - Gain knowledge, make decisions
  - Build runway for future Features

- Implement
  - Build Features in the Roadmap



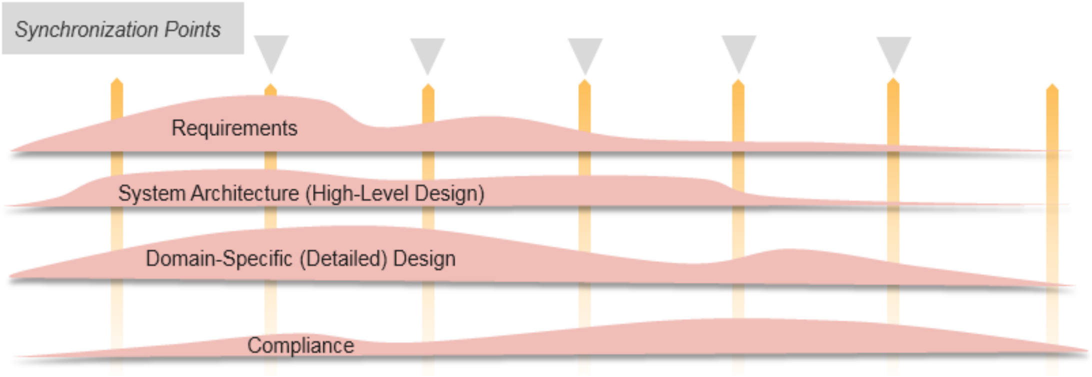
Emergent Specifications

- Sequential specifications
  - Delay feedback on decisions
  - False sense of product maturity
  - Creates large batches of work



• Handoffs
• Large batches
• Wishful Thinking
Requirements
System Architecture (High-Level Design)
Domain-Specific (Detailed) Design

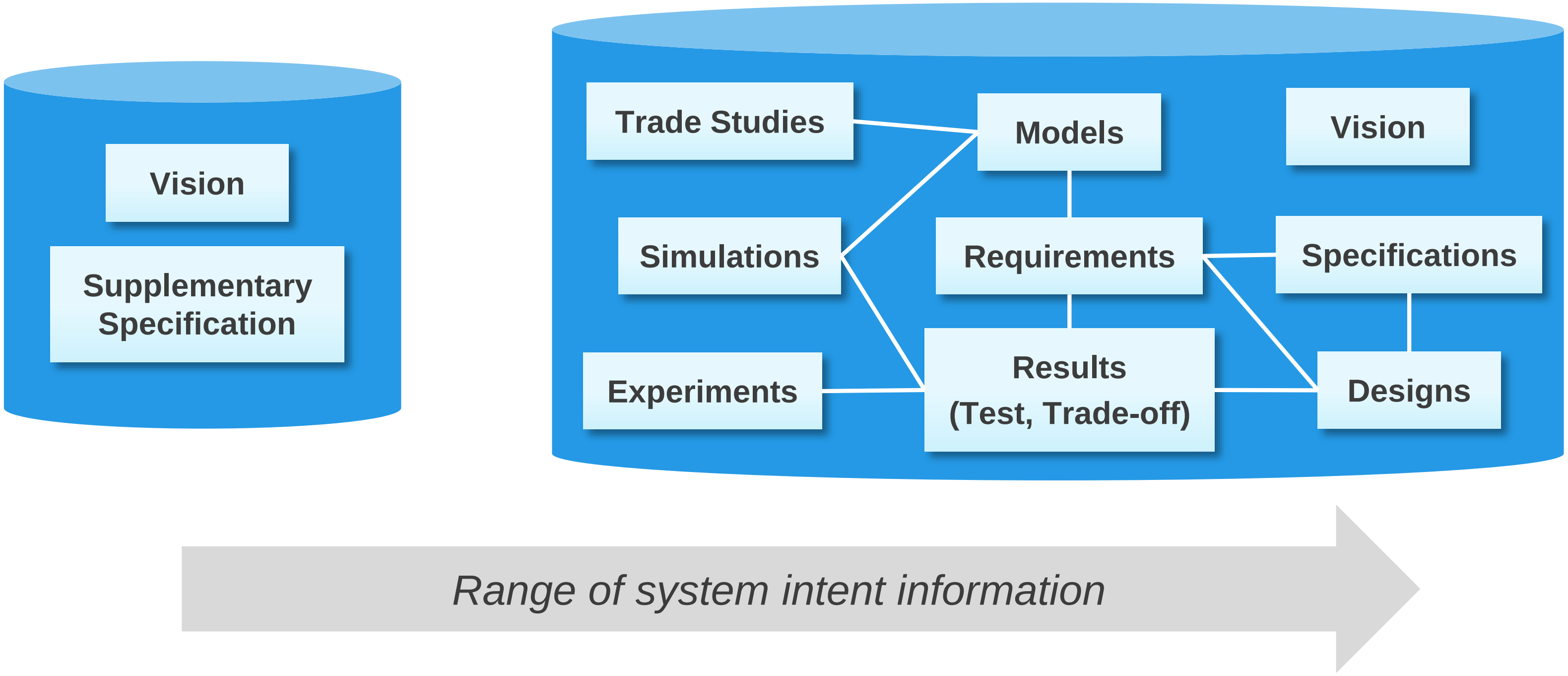
- Concurrent Specifications
  - Developed in parallel
  - Can begin understood work sooner
  - Provides for tradeoffs
  - Smaller batches



Synchronization Points
Requirements
System Architecture (High-Level Design)
Domain-Specific (Detailed) Design
Compliance

System Intent

- Current statement of *what we are building* and *how we are building it*
- Align system builders and guide their implementation
- Facilitate collaboration with system, subsystems, suppliers
- Support compliance and contractual obligations

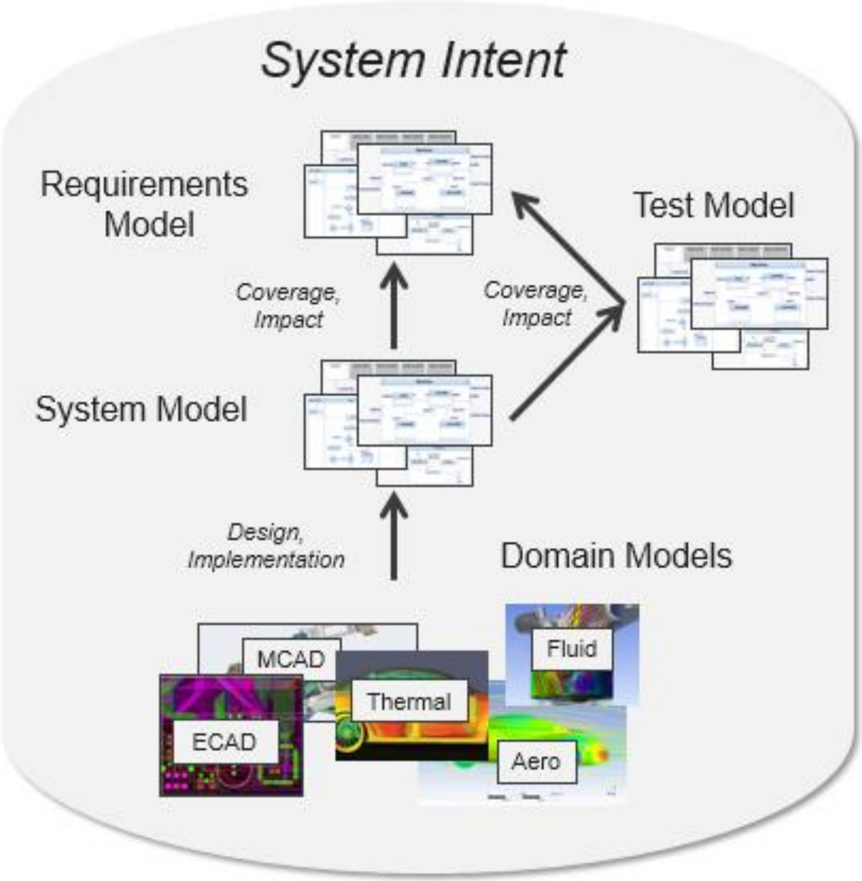

Vision
Supplementary Specification
Trade Studies
Models
Vision
Simulations
Requirements
Specifications
Experiments
Results (Test, Trade-off)
Designs
Range of system intent information

System Intent - Minimal but Sufficient

- Favor models over documents
- Everyone contributes
  - Not an *Ivory Tower;* No monopoly on ideas and innovation
- Keep options open
  - Explore options as long as they are economically feasible
  - Defer decisions where possible
- System intent is the “single source of truth”
- Keep it high level; keep it simple
  - Intent is means to an end

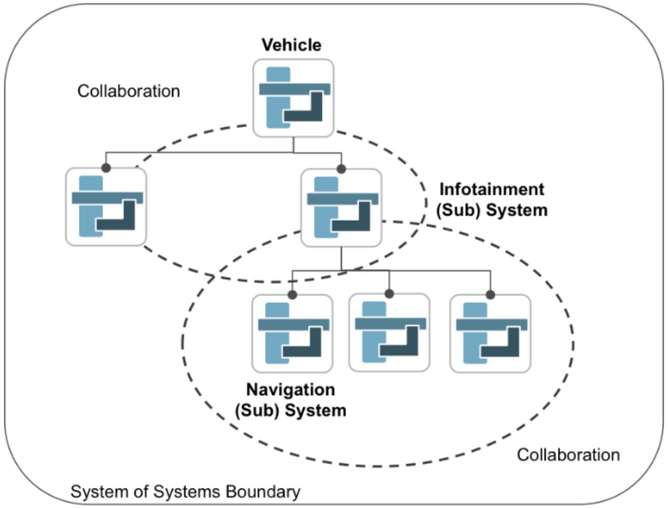
Model Based Systems Engineering (MBSE)

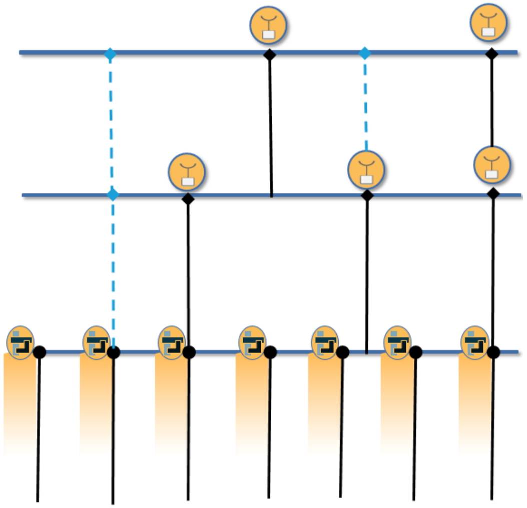
- Favor information in models over documents
  - Model – granular, attributes, links, query, visualize, report, generate documents
  - <u>A document repository is not a model ☺</u>



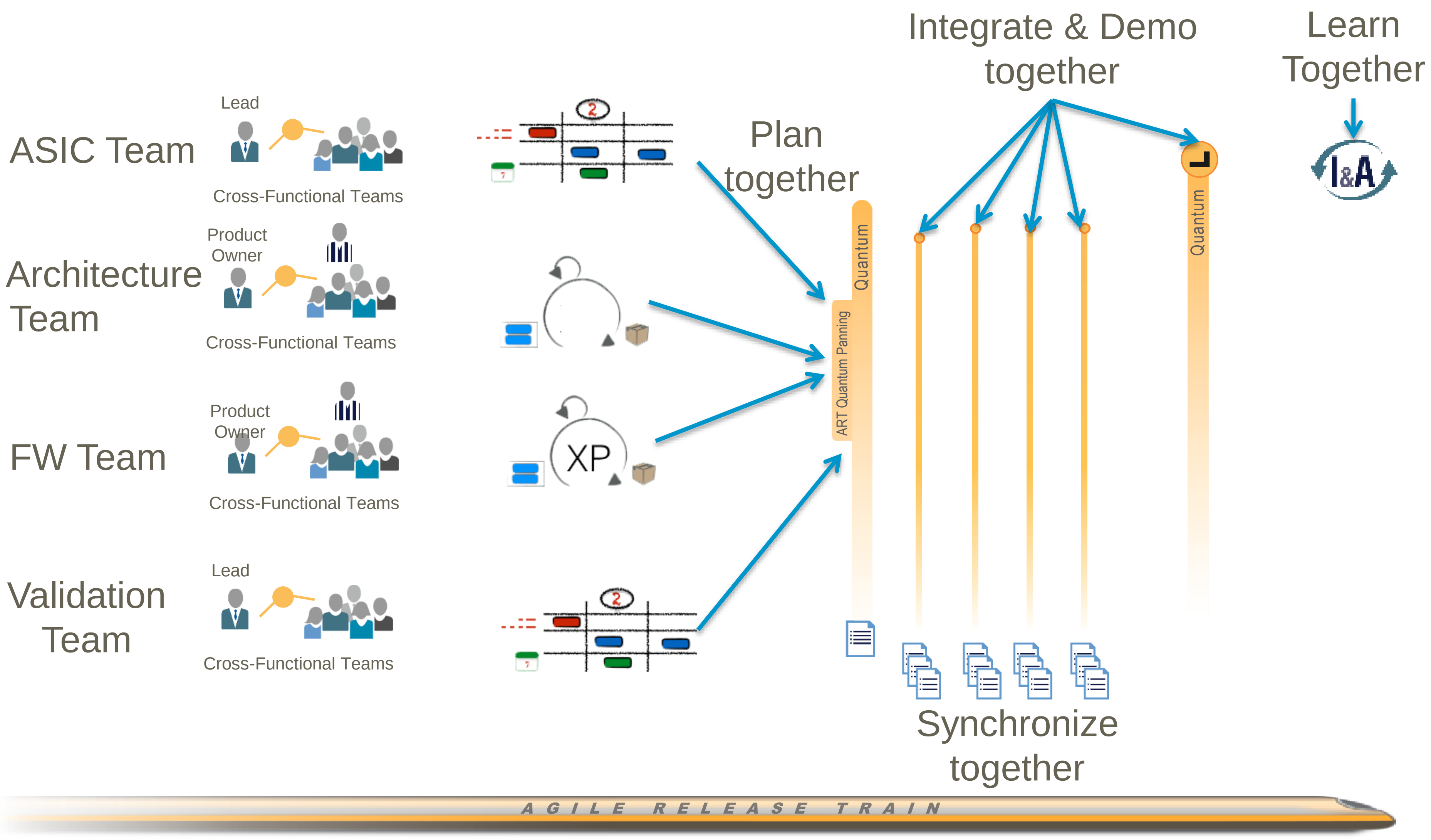
System of Systems - Suppliers

- “The whole is more than the sum of its parts”
- System of system is collection of collaboration groups
  - Plan, integrate, test, demo as a group
- Integration points control product development
  - Large typically implies less frequent
  - Not all integrations require physical production (*soft integration*)





Different Teams, Processes. One Train, Direction



ASIC Team

Architecture Team

FW Team

Validation Team

Plan together

Integrate & Demo together

Learn Together

Synchronize together

Harry Koehnemann

Director of Technology

harry@321gang.com