Using Systems Engineering Workshops and MBSE to drive agile development

Harry Koehnemann, Ph.D.
SAFe Fellow, Principle Consultant
Scaled Agile Inc.

Agenda

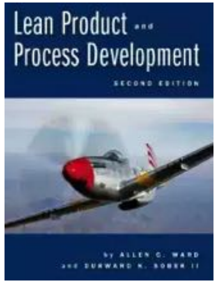
- Brief introduction to Lean and Agile
- Make SE activities part of Agile’s flow of work
- Agile SE Workshop basics - Analysis
- Agile SE Workshop basics - Design

Brief introduction to Lean and Agile

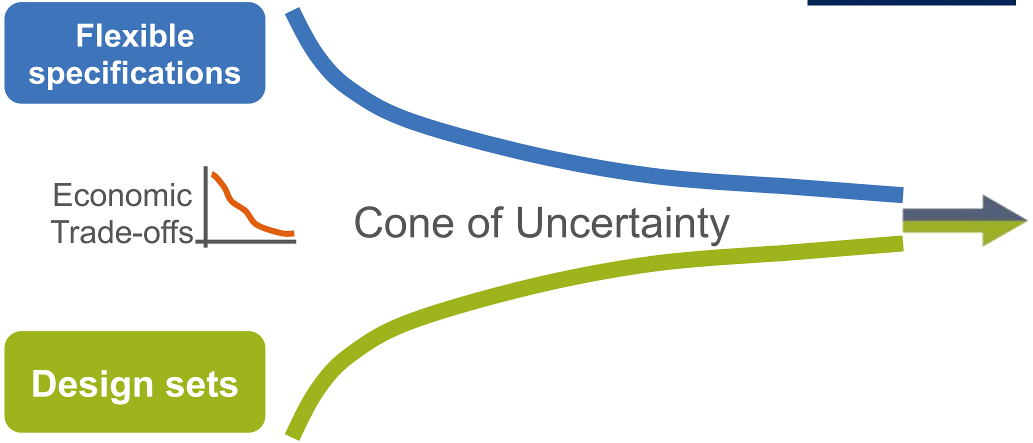
Assume variability; preserve options

Aggressively evaluate alternatives. Converge specifications and solution set.

—Allen Ward


Lean Product and Process Development
SECOND EDITION
by ALLEN C. WARD and DURWARD K. SOBEK II

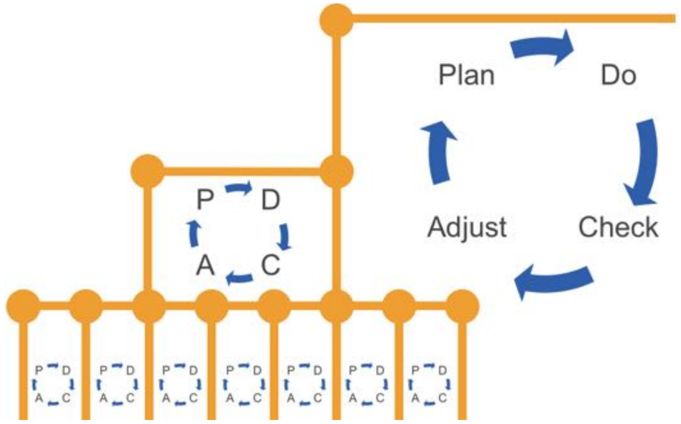
- We cannot possibly know everything at the start
- Requirements must be flexible to make optimal economic design choices
- Designs must be flexible to support changing requirements
- Preservation of options improves economic results

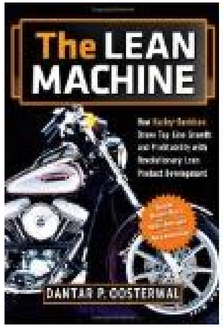


Build incrementally with fast, integrated learning cycles

Integration points control product development.
— Dantar P. Oosterwal

- Integration points accelerate learning
- Development can proceed no faster than the slowest learning loop
- Improvement comes through synchronization of design loops and faster learning cycles

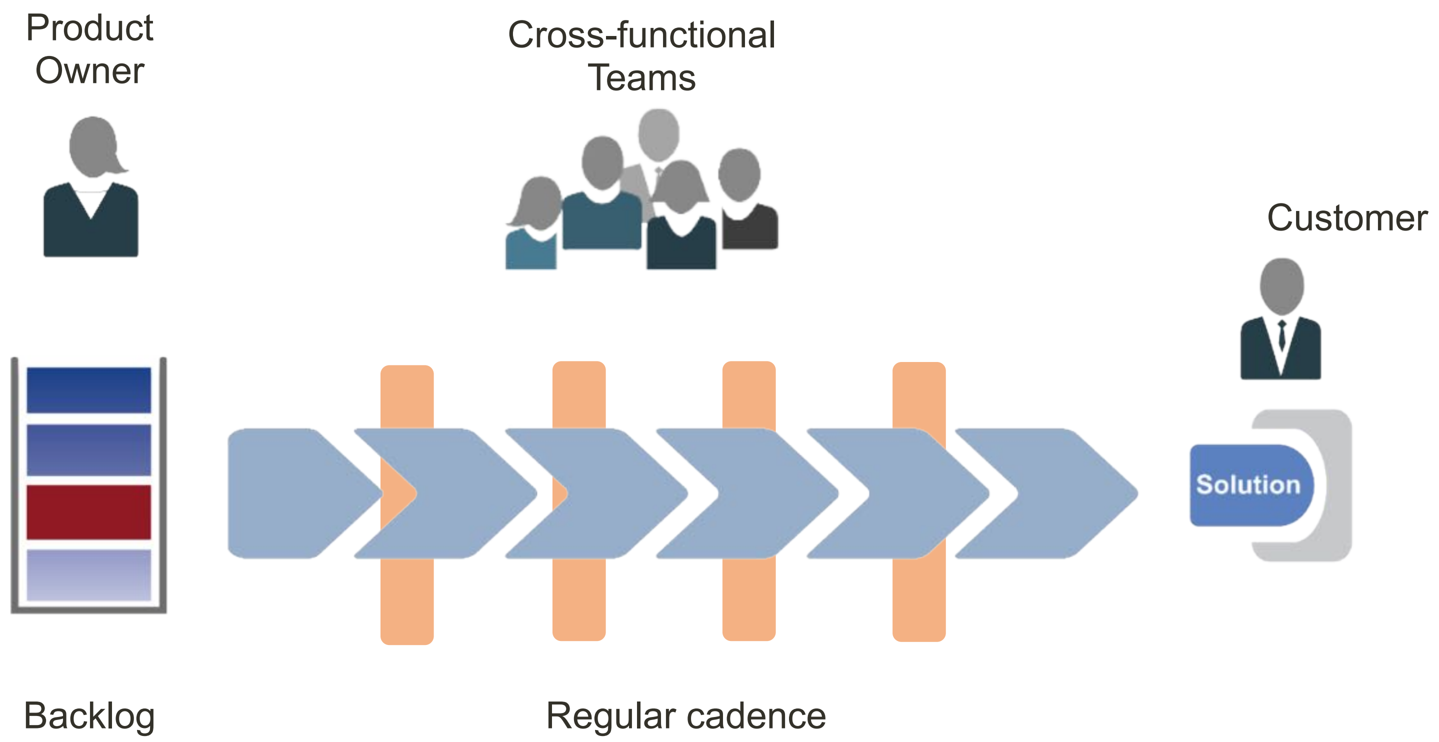




The Lean Machine*:*
How Harley Davidson Drove Top-Line Growth and Profitability with Revolutionary Lean Product Development

—Dantar P. Oosterwal

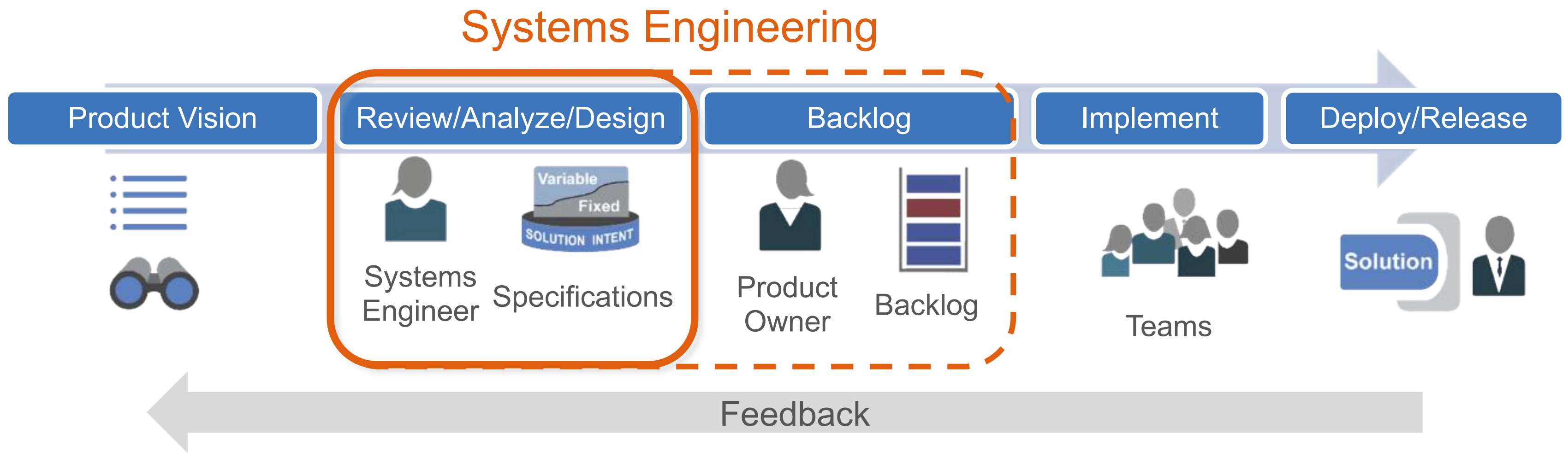
Agile/Scrum basics



Make SE activities part of Agile's flow of work

Make Systems Engineering work part of agile flow

- In Lean-Agile, all work is flow-based and performed in small batches
- Consequently, SE activities must be part of flow

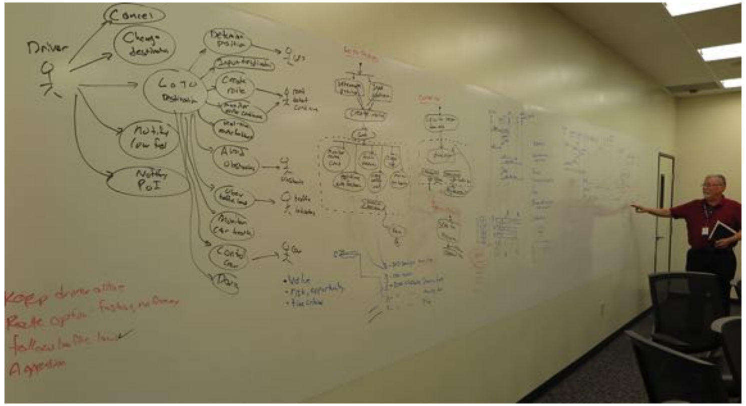


Use workshops to drive backlogs

The most efficient and effective method of conveying information is face-to-face conversation

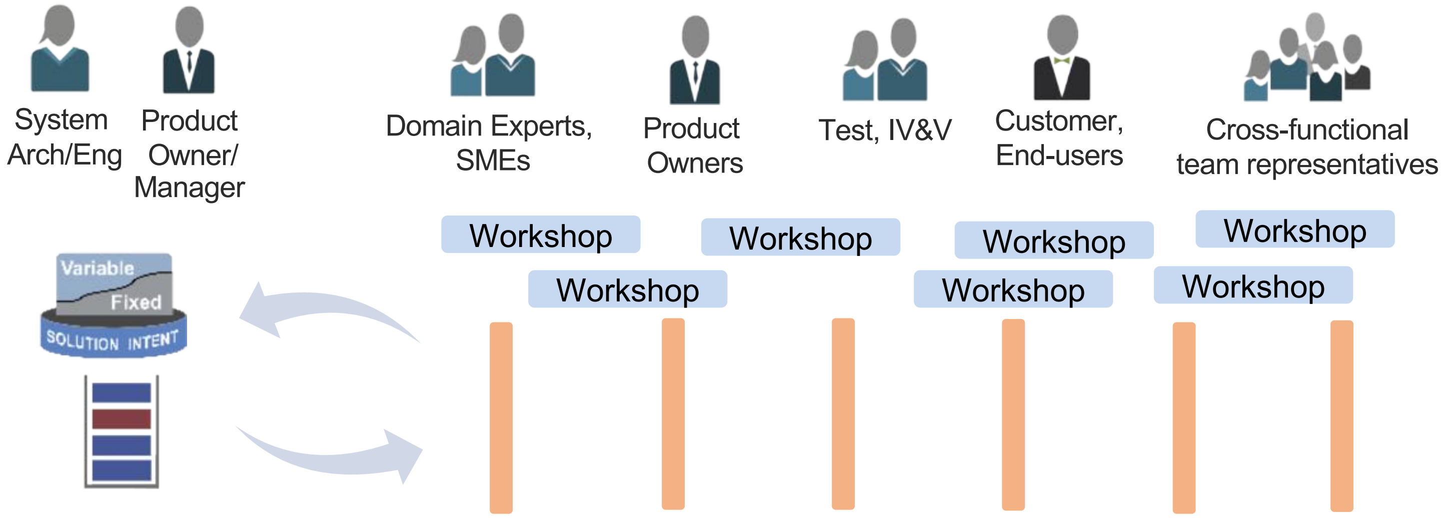
-- Agile Manifesto Principle

- To be relevant in an agile context, Systems Engineering (SE) efforts must continually influence and guide the work in backlogs
- Based Agile's *Specification Workshop* which are face-to-face meeting to:
  - Discuss questions about system behavior
  - Identify gaps in the backlog
  - Align everyone on how the system will work



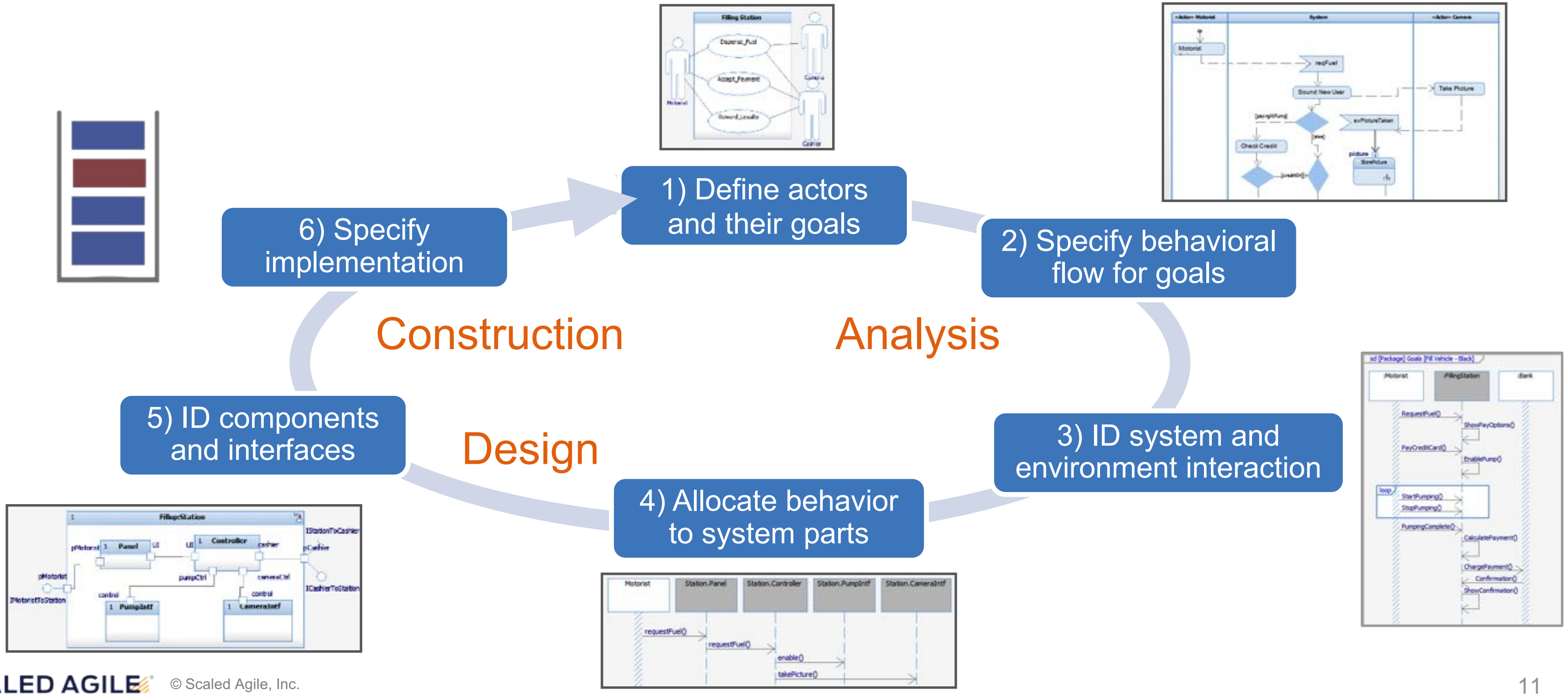
Many workshops performed continually throughout lifecycle

- Workshops have different focus – and therefore different attendees
  - Analysis – scope, behavior
  - Design – interactions, interfaces, allocations



Workshops apply the same Systems Engineering activities

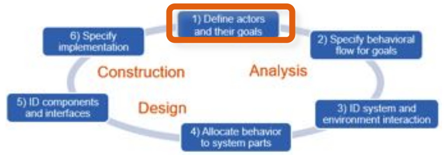
- But! Performed continuously and in (much) smaller batches

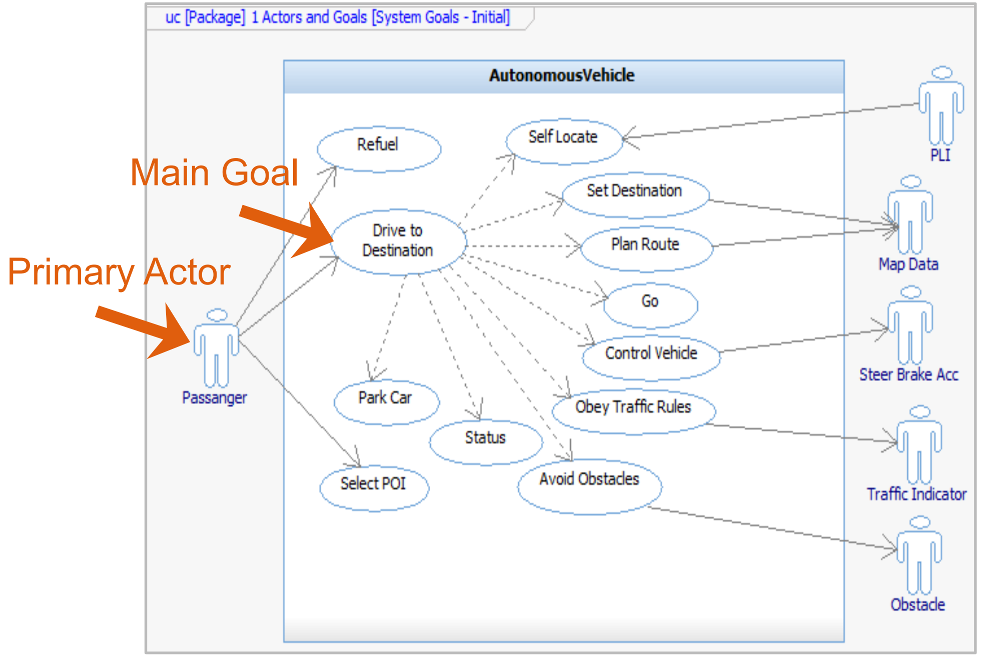


Agile SE Workshop basics - Analysis

1) Apply Use Case modeling – actors and their goals

- Start with highest business value, which is commonly system's primary behavior
- Find the system's primary actor and that actor's main goal (the *Alpha Thread*)
- What are sub-goals of the main goal?
- What actors do they interface?

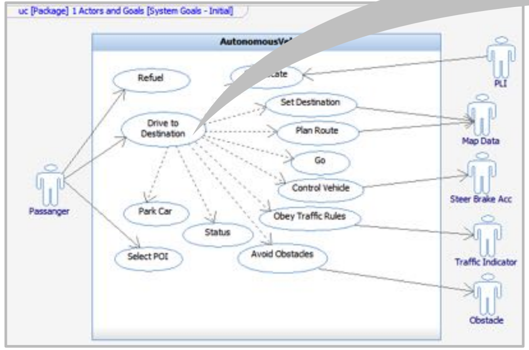

1) Define actors and their goals
2) Specify behavioral flow for goals
3) ID system and environment interaction
4) Allocate behavior to system parts
5) ID components and interfaces
6) Specify implementation
Analysis
Design
Construction

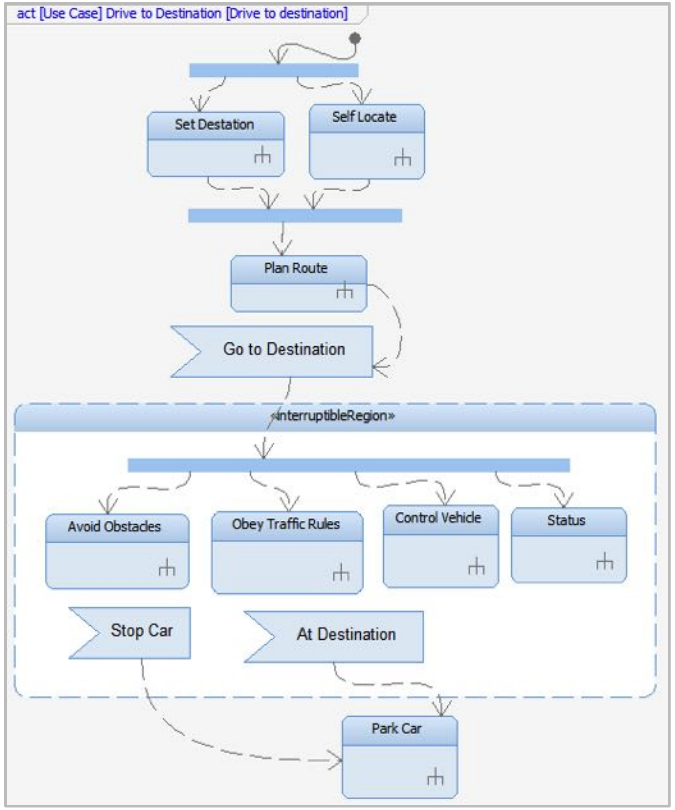


2a) Specify behavioral flow for goals

1) Define actors and their goals
2) Specify behavioral flow for goals
3) ID system and environment interaction
4) Allocate behavior to system parts
5) ID components and interfaces
6) Specify implementation
Analysis
Design
Construction

- Describe behavior as an activity of sub-goals
- Defining the thread reveals additional sub-goals and actors
- Initially, stay focused on minimal, viable behavior

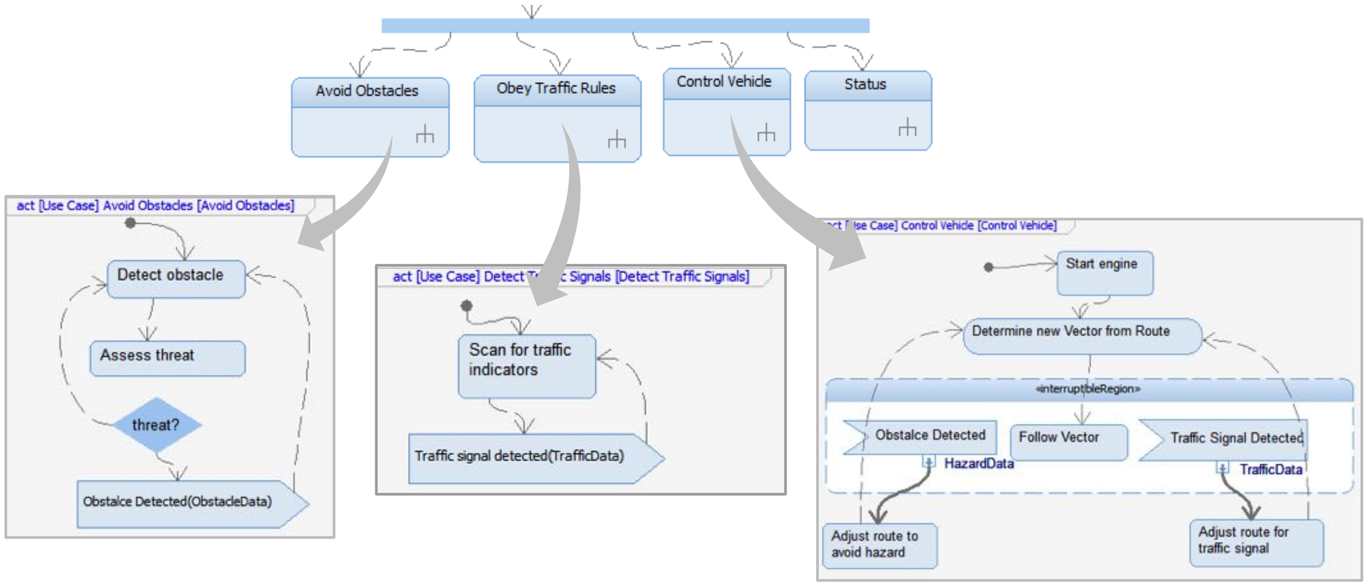




2b) Specify behavioral flow for goals

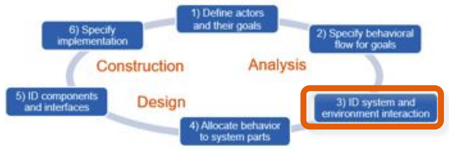
1) Define actors and their goals
2) Specify behavioral flow for goals
3) ID system and environment interaction
4) Allocate behavior to system parts
5) ID components and interfaces
6) Specify implementation
Construction
Analysis
Design

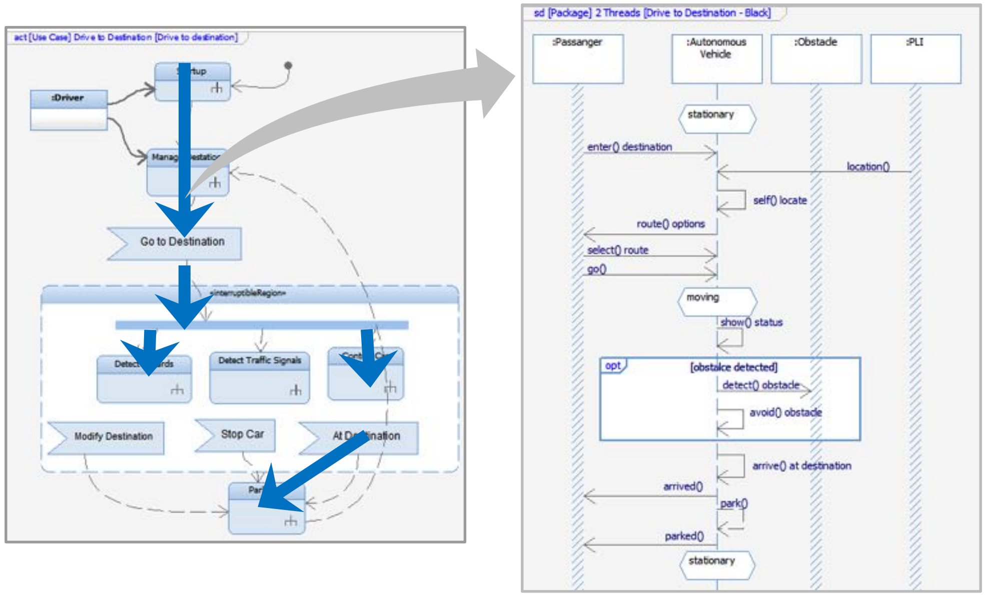
- Repeat the process for sub-goals
- In general, do not model below sub-goals


Avoid Obstacles
Obey Traffic Rules
Control Vehicle
Status
act [Use Case] Avoid Obstacles [Avoid Obstacles]
Detect obstacle
Assess threat
threat?
Obstalce Detected(ObstacleData)
act [Use Case] Detect Traffic Signals [Detect Traffic Signals]
Scan for traffic indicators
Traffic signal detected(TrafficData)
act [Use Case] Control Vehicle [Control Vehicle]
Start engine
Determine new Vector from Route
«interruptibleRegion»
Obstalce Detected
Follow Vector
Traffic Signal Detected
HazardData
TrafficData
Adjust route to avoid hazard
Adjust route for traffic signal

3) Create system threads from goals and sub-goals

- Create black box ("skinny") interaction flow for each Actor Goal
- Define the stimulus in and out of the system as a sequence of events
- System is a black box, no internal components
- Useful for system-level tests


1) Define actors and their goals
2) Specify behavioral flow for goals
3) ID system and environment interaction
4) Allocate behavior to system parts
5) ID components and interfaces
6) Specify implementation
Analysis
Design
Construction



Consider threads as Agile Epics

- Epics are large, cross-cutting initiatives that deliver significant business value
- Strive for minimal viable Epics – small batches of value
- Each activity will have many Threads/Epics of varying scope

Epic: Stay in lane

Epic: Detect side obstacles

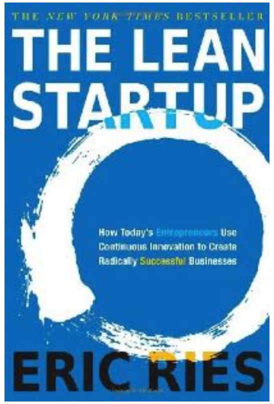
Epic: Obey Traffic Lights

Epic: Simple drive to destination

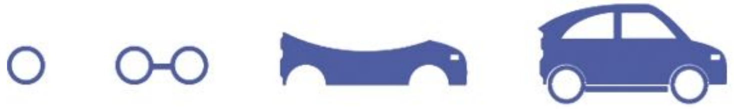
Epic: Detect forward vehicle

What is a good Epic for complex systems?

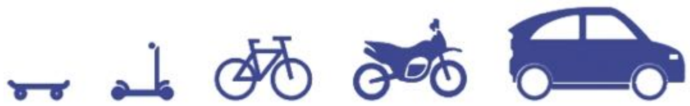
- Validate assumptions early
- Favor early, end-to-end capability (Alpha Thread)
- Follows Eric Reis' Minimum Viable Product (MVP)
  - Build minimum solution to gain desired knowledge


THE NEW YORK TIMES BESTSELLER
THE LEAN STARTUP
How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses
ERIC RIES

Engineering by building parts



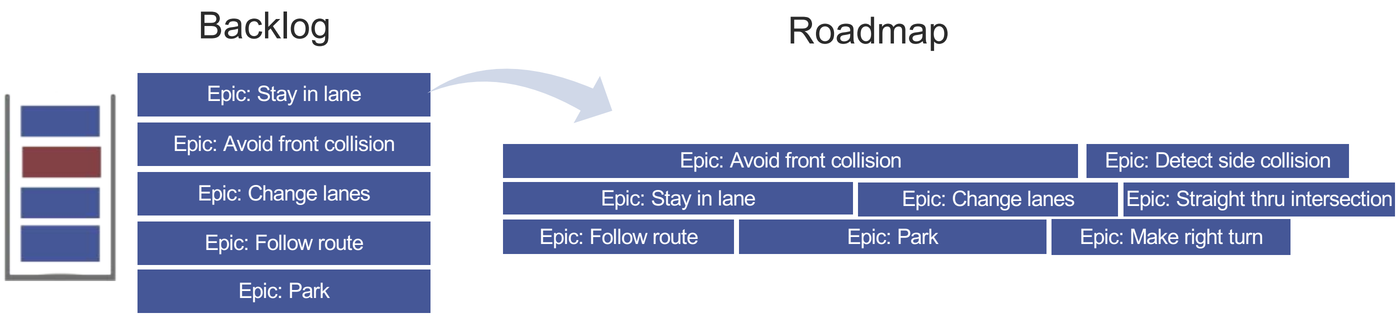
Engineering by demonstrable learning



Prioritize Epics based on Cost of Delay (Reinertsen)

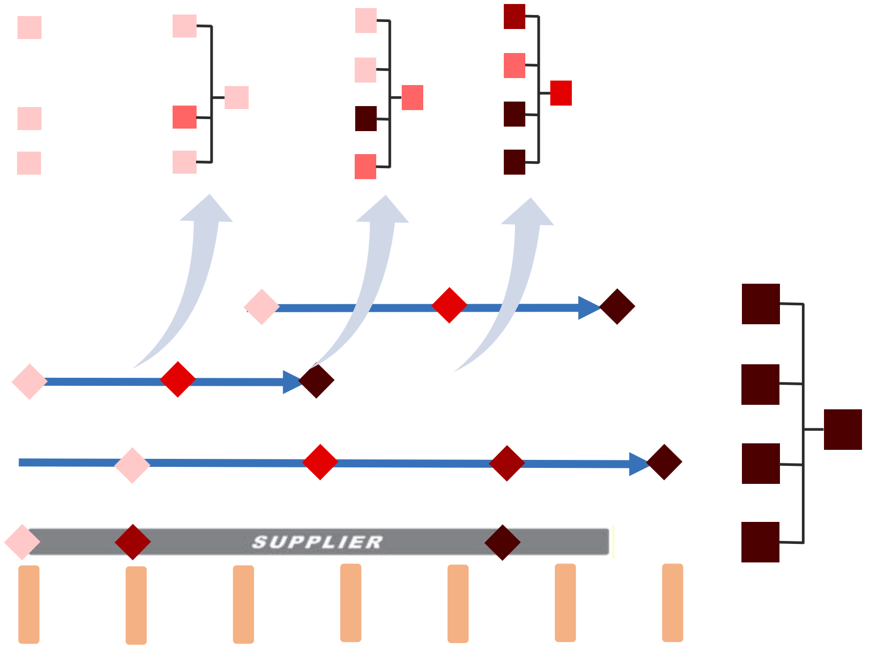
- Understand the cost of *not* doing something
- SAFe's Weighted Shortest Job First (WSFJ) is a proxy for CoD:

$$\text{WSJF} = \frac{\text{CoD}}{\text{Size}} = \frac{\text{User-business value} + \text{Time criticality} + \text{Risk Reduction} \mid \text{Opportunity Enablement}}{\text{Size}}$$



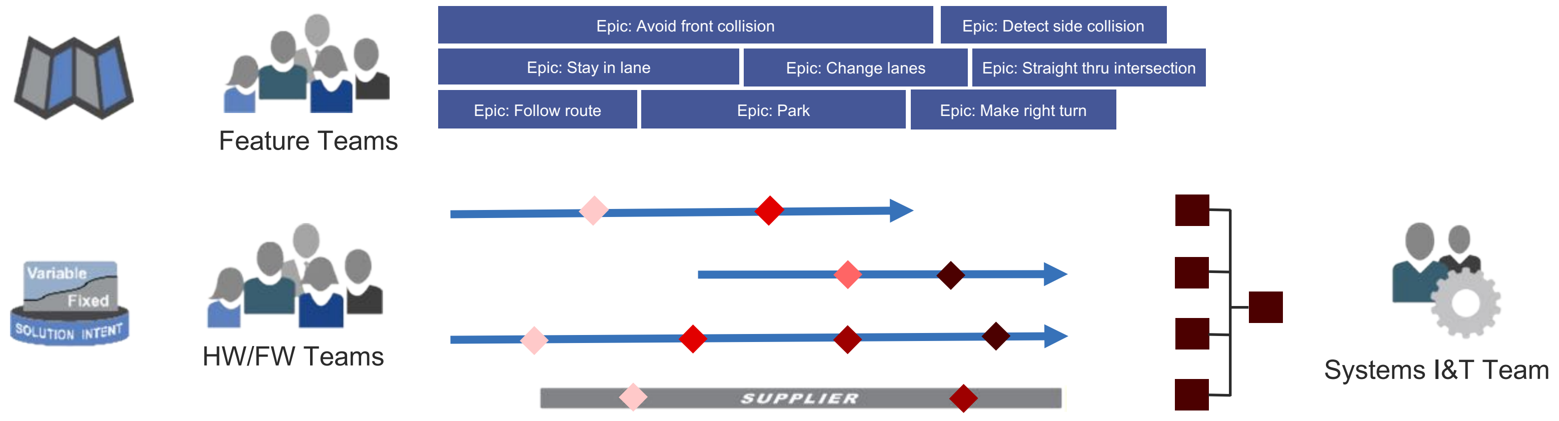
Define component roadmap

- Leverage prototype incremental solutions
- Strive for end-to-end solution early for faster validation feedback
- Bring production in early to validate manufacturability
- Make sufficient testing platforms available for teams to integrate and test their parts of the solution



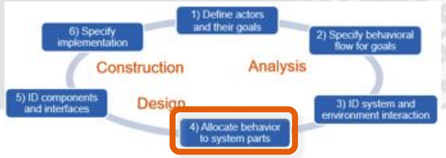
Align functional and component roadmaps

- What component capabilities are required to implement Epics
- Will the technology perform as we anticipate

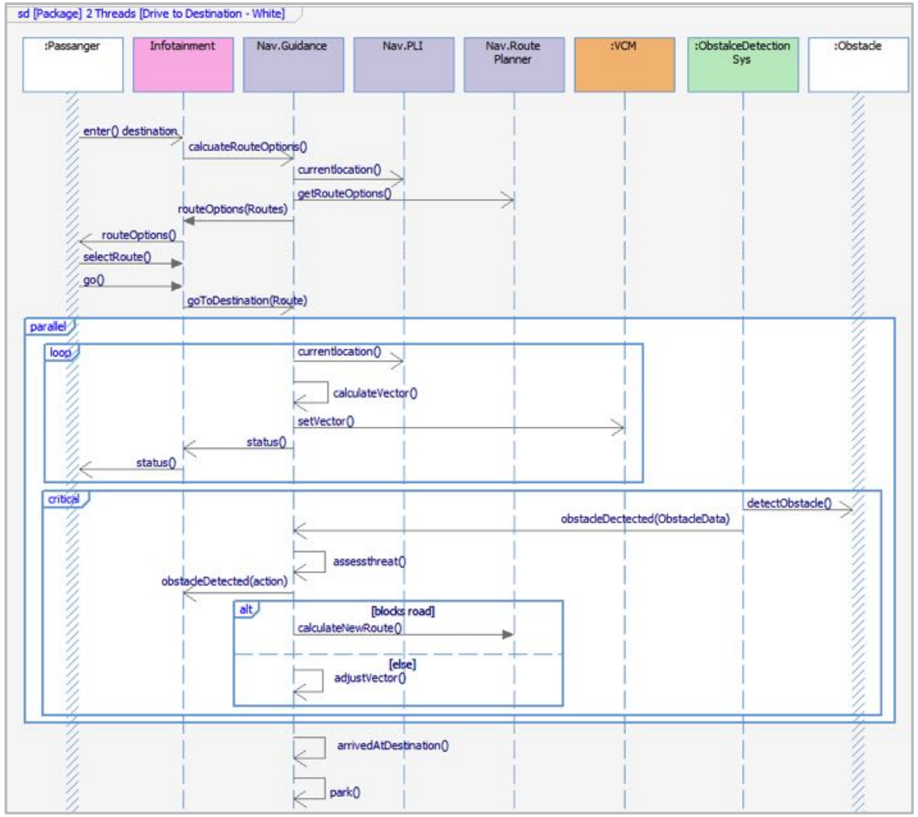


Agile SE Workshop basics - Design

4) Allocate behavior to system parts


1) Define actors and their goals
2) Specify behavioral flow for goals
3) ID system and environment interaction
4) Allocate behavior to system parts
5) ID components and interfaces
6) Specify implementation
Construction
Analysis
Design

- For each external interaction, decide how system elements realize the behavior
- First step in design - determines how behavior is realized by system elements
- Discovers system parts and allocates responsibilities to them

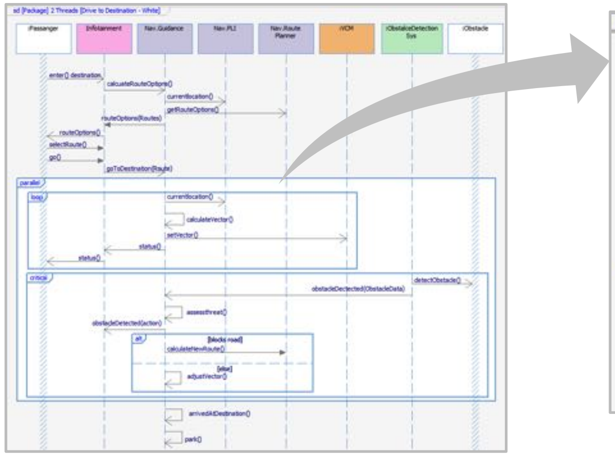


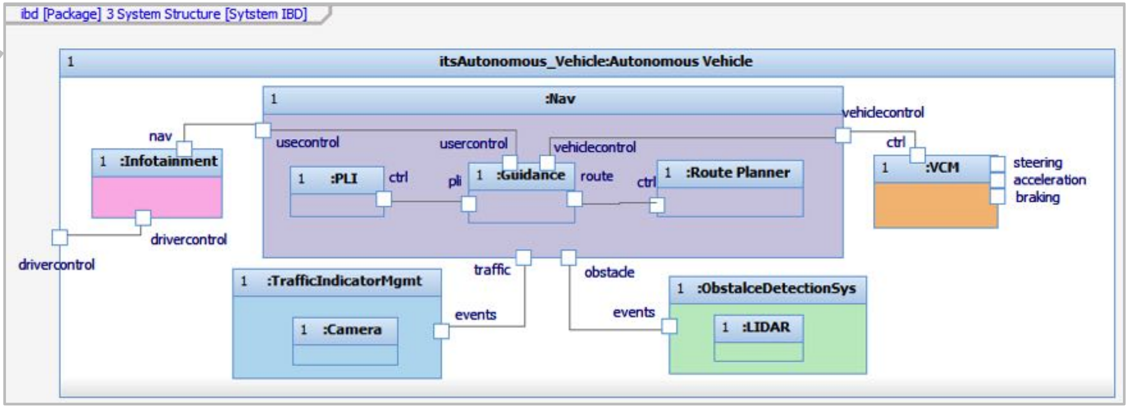
5) Id systems components and interfaces

- Each interaction requires an interface

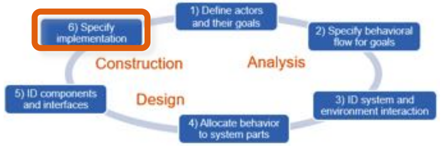
1) Define actors and their goals
2) Specify behavioral flow for goals
3) ID system and environment interaction
4) Allocate behavior to system parts
5) ID components and interfaces
6) Specify implementation
Analysis
Design
Construction

- Use behavior to drive interface specs between system parts





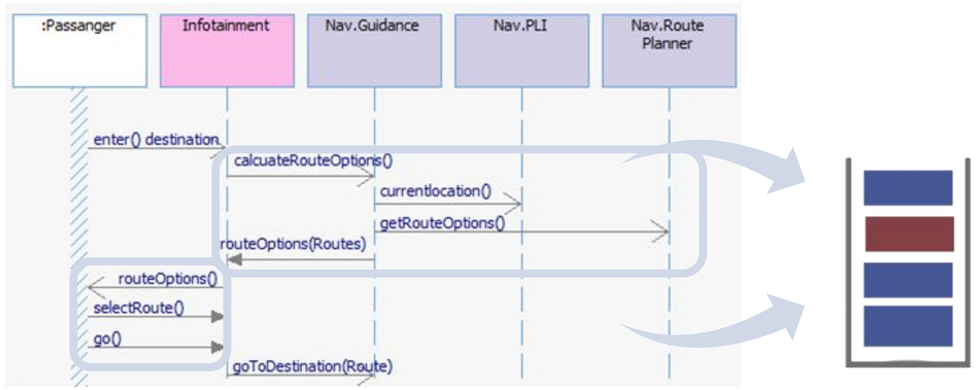
6) Define backlog items from interactions


1) Define actors and their goals
2) Specify behavioral flow for goals
3) ID system and environment interaction
4) Allocate behavior to system parts
5) ID components and interfaces
6) Specify implementation
Analysis
Design
Construction

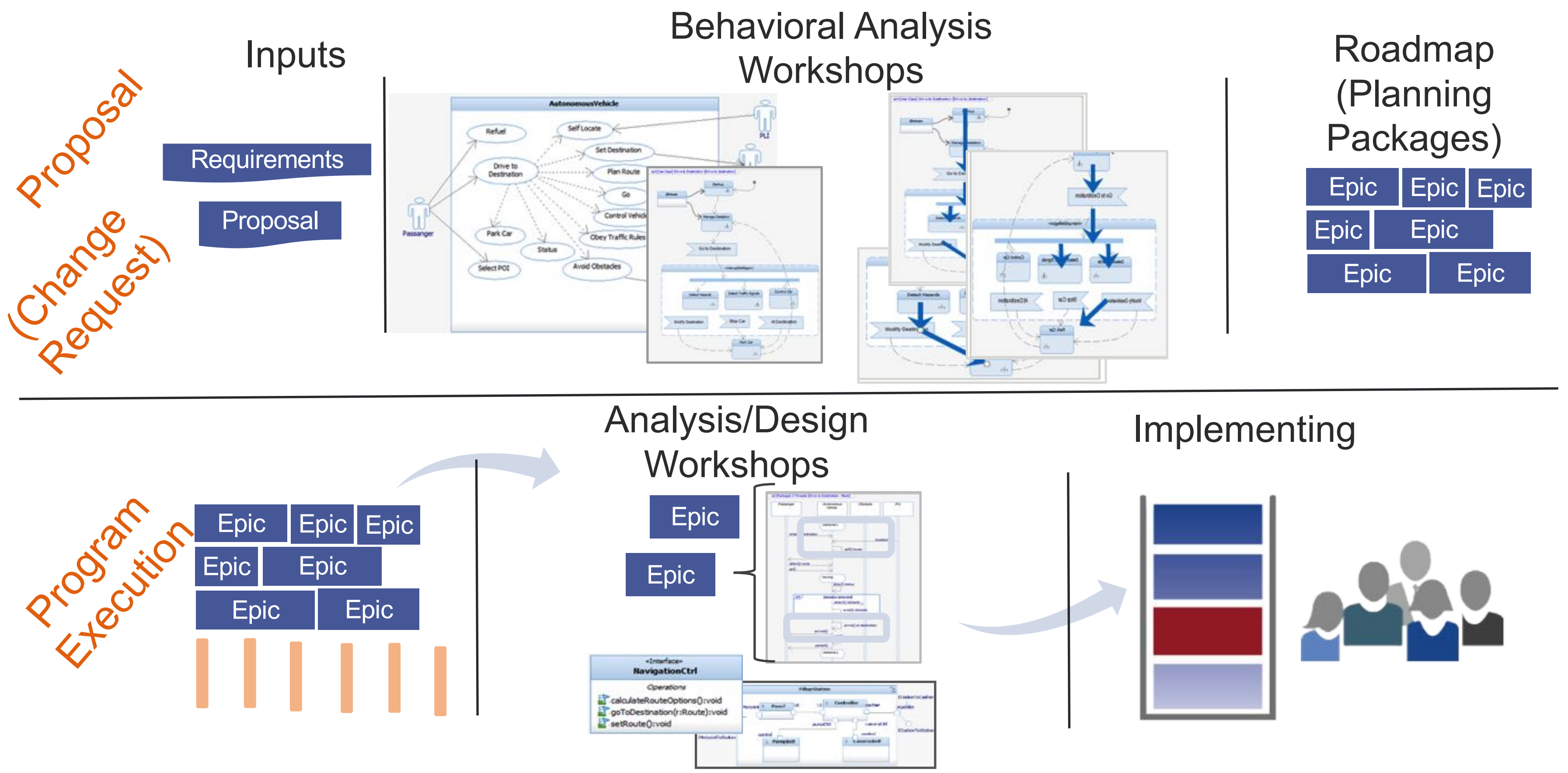
- Discover Features from groups of interactions
- Some Features may require exploration work
- Focus on requests from UI and controller parts of system

As Infotainment, I want a set of route options so that the user can select the optimum route from their current location

As a passenger, I want to select my route so that I can decide the optimal route to travel to my destination

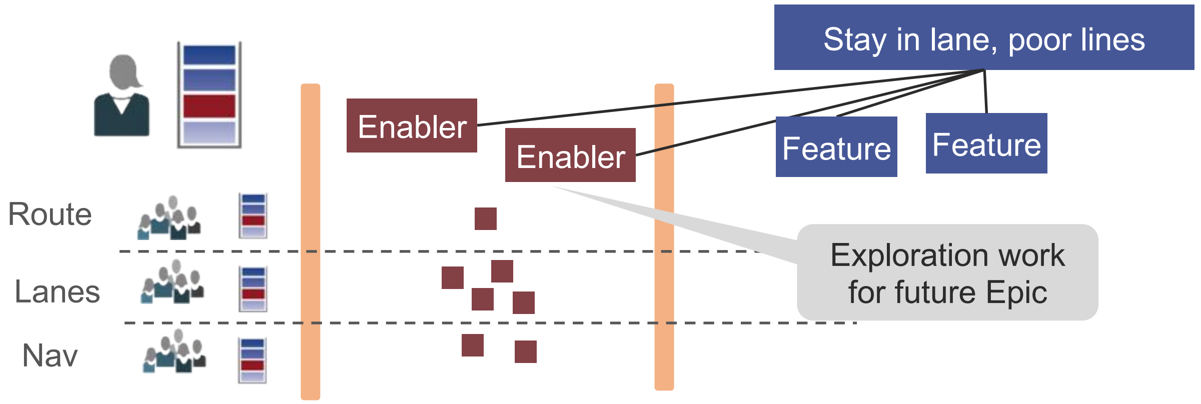


Applying Workshops in a contracted (gov) environment



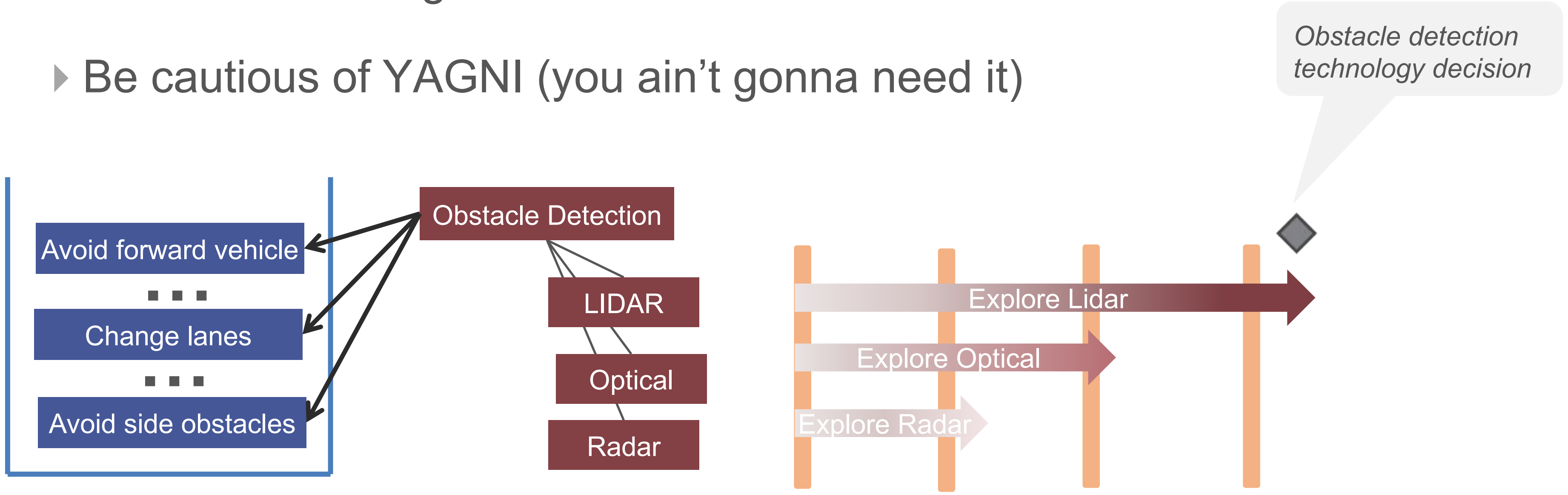
Ensure sufficient knowledge to implement Epics

- May explore Epics prior to implementation – design alternative, trade studies, etc.
- Ensure teams can commit to Epic work during planning
- Continuously perform analysis and design work



Exploration may consider requirements for future Epics

- Economic trade-off between additional effort, longer decision cycle vs. later cost of change
- Be cautious of YAGNI (you ain't gonna need it)



Change our approach to systems engineering

Old approach		New approach
Define detailed, up-front specifications	→	Support emergent specifications; adapt based on new knowledge
Drive teams to meet detailed, fixed schedule	→	Own vision and roadmap; allow teams flexibility in implementation
Hand off detailed specifications and plans to teams	→	Understand face-to-face is best form of communication
Technical leader, primary decision maker, bottleneck for questions	→	Create the environment for teams to explore and learn

Thank you!

Harry Koehnemann
harry.koehnemann@scaledagile.com