



SAFe with MBSE for System Engineers



WHO WE ARE.



Peter Luckey

321 Gang helps organizations improve their ability to design and develop high- quality systems and software. Our team of experts will help you in adopting the best agile/lean or traditional practices and/or tools for your organization's unique challenges and ROI goals.

We are an IBM Platinum Business Partner, Authorized IBM Watson IOT reseller, and IBM Certified Training Partner- providing our clients a 'one stop shopping' experience.

Premise

It is important to involve System Engineering in the identification of the features to be developed during each Program Increment (PI).

Model Based System Engineering (MBSE) enables the System Engineers to:

- ✓ Collaboratively create a shared understanding of system structure, behavior, and interfaces
- ✓ Record decisions as a single source of truth using standard notation
- ✓ Use emergent specifications to drive the agile development process



Today's agenda



Presentation
40 minutes



Q&A
10 minutes

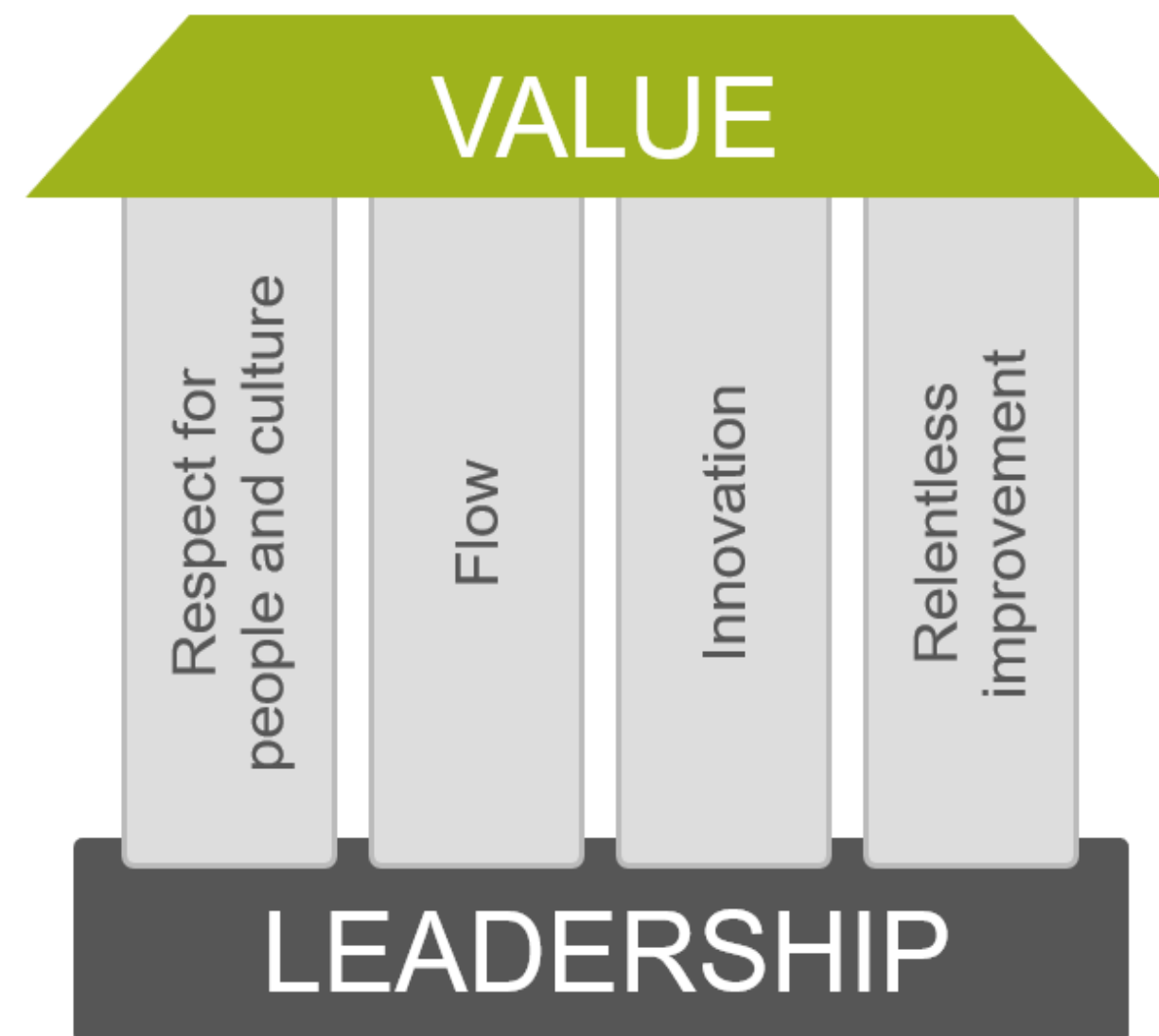
- Understand Lean-Agile principles, practices, and roles
- Know the characteristics of a specification workshop
- Detail a Systems Engineering workshop
 - Part 1 – Analysis
 - Part 2 – Design / Construction
- Making effective change



Understand Lean-Agile principles, practices, and roles

Lean-Agile Foundations

SAFe House of Lean



SCALED AGILE © Scaled Agile, Inc.

2.5

The Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

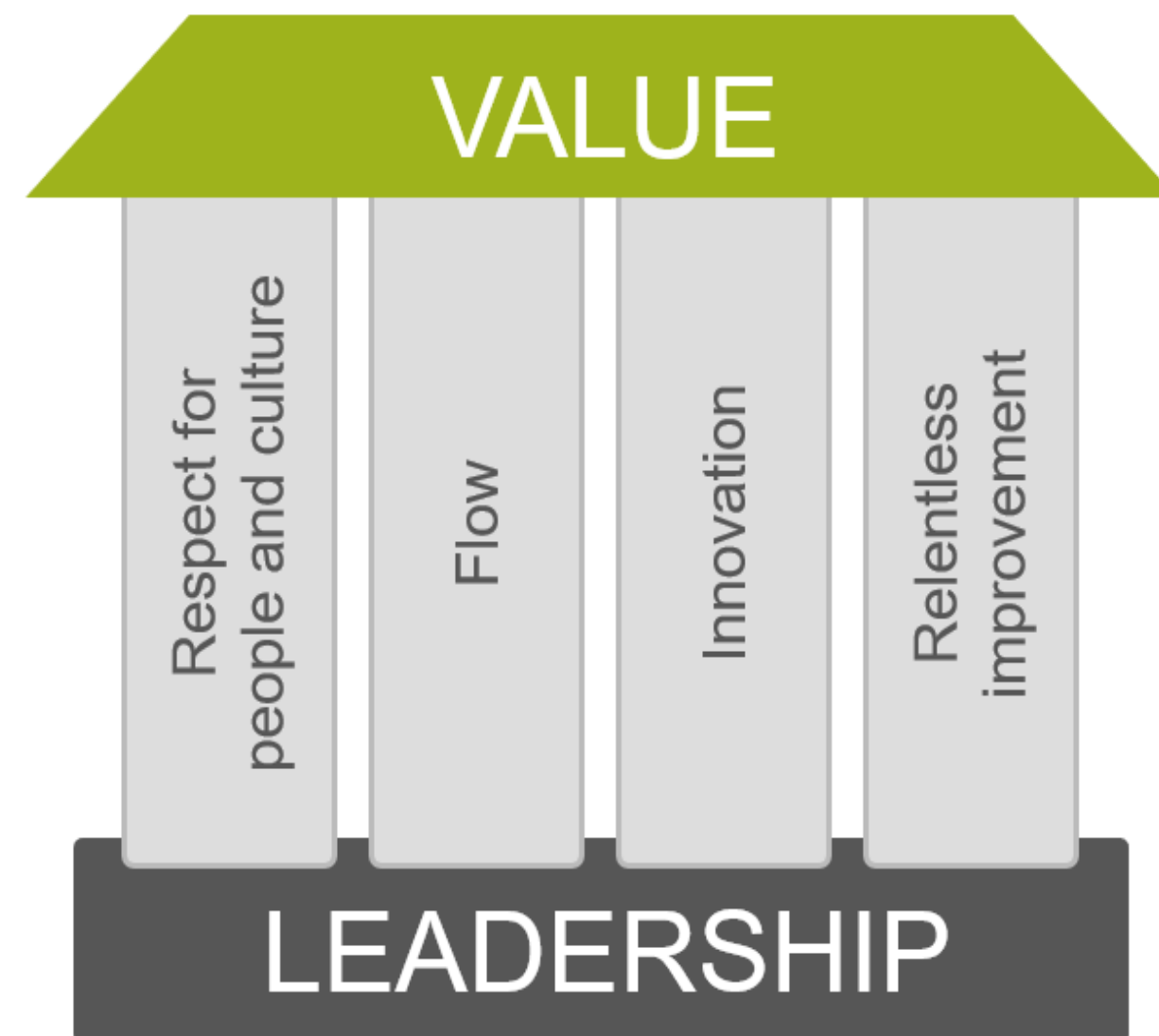
 agilemanifesto.org

SCALED AGILE © Scaled Agile, Inc.

2.15

Lean-Agile Foundations

SAFe House of Lean



SCALED AGILE © Scaled Agile, Inc.

2.5

The Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

 agilemanifesto.org

SCALED AGILE © Scaled Agile, Inc.

2.15

SAFe Lean-Agile principles

#1 - Take an economic view

#2 - Apply systems thinking

#3 - Assume variability; preserve options

#4 - Build incrementally with fast, integrated learning cycles

#5 - Base milestones on objective evaluation of working systems

#6 - Visualize and limit WIP, reduce batch sizes, and manage queue lengths

#7 - Apply cadence, synchronize with cross-domain planning

#8 - Unlock the intrinsic motivation of knowledge workers

#9 - Decentralize decision-making

#10 – Organize around value

Agile Teams

➤ Product Owner

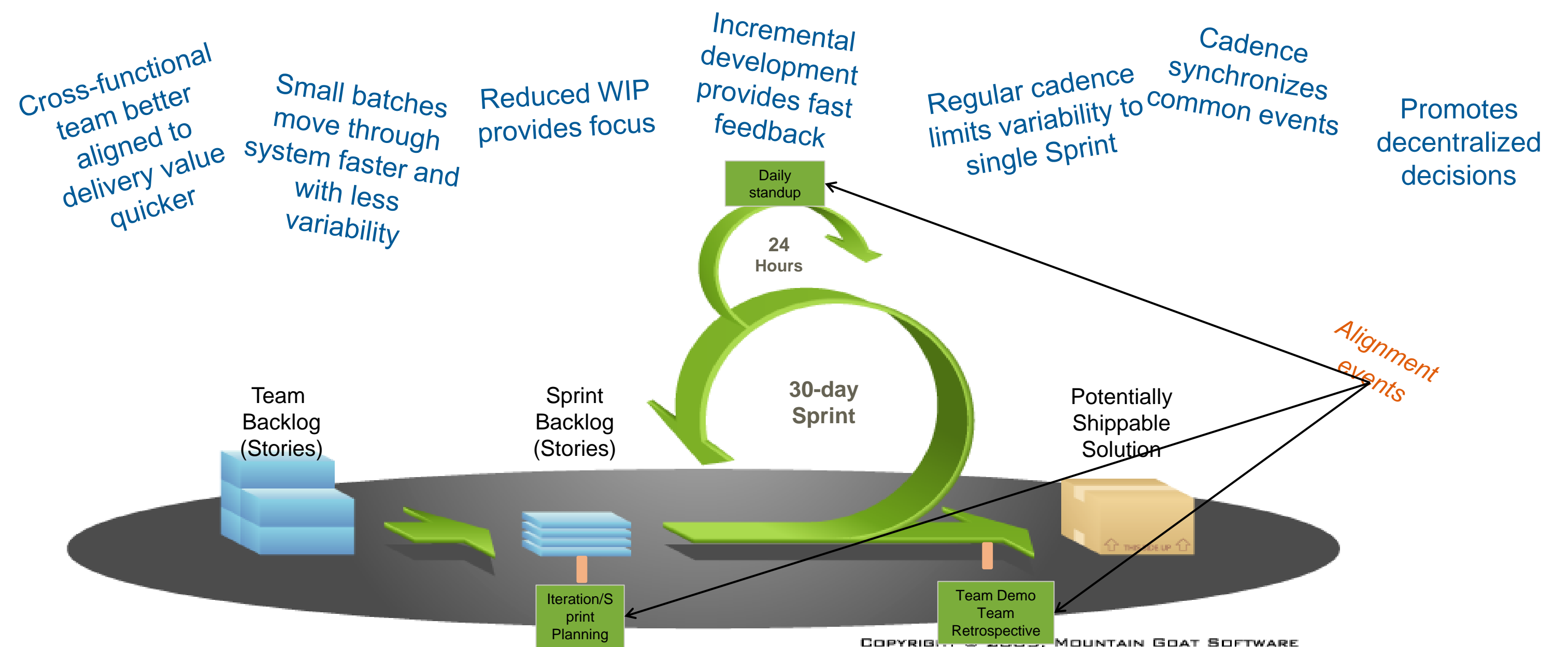
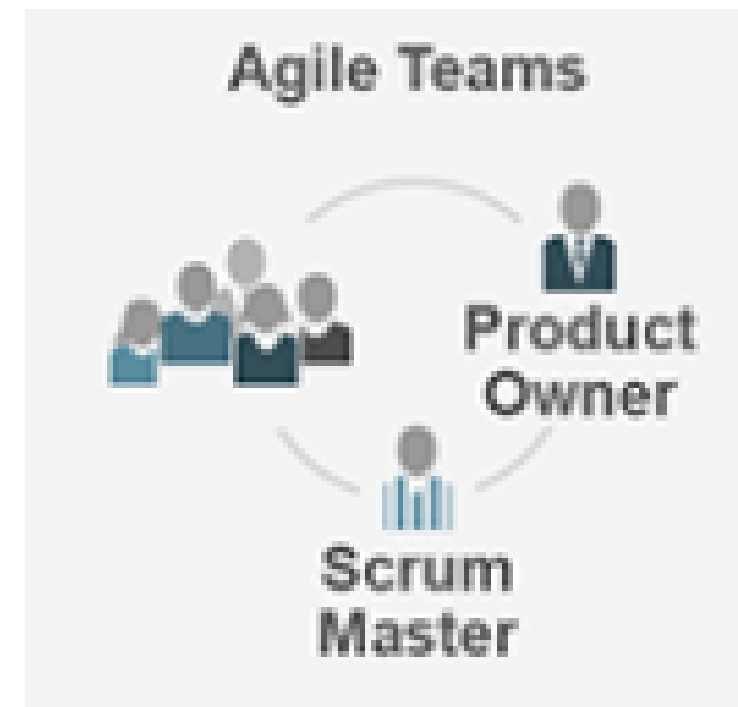
- Serves as customer proxy to the team
- Owns vision, and team's backlog

➤ Scrum Master

- Helps team achieve its goals; remove impediments
- Coaches team on lean-agile practices

➤ Other Team Members

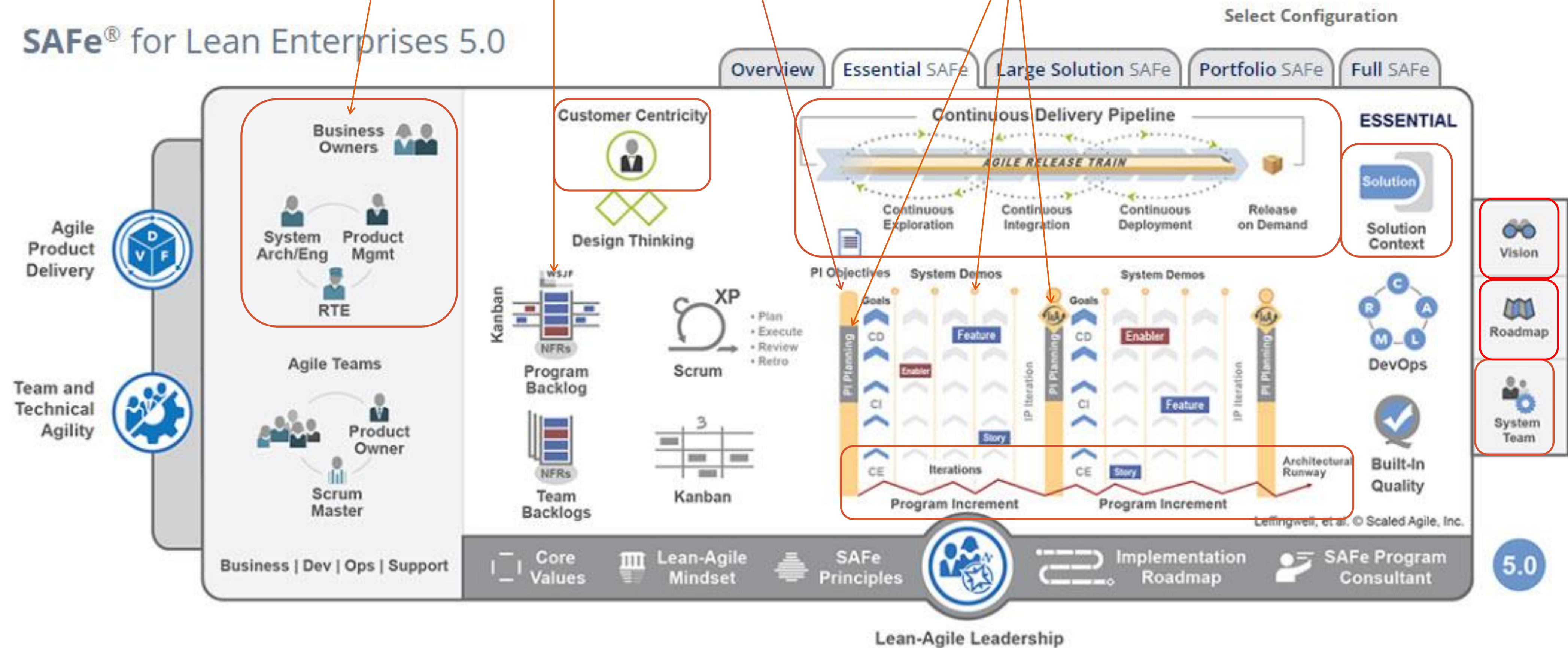
- Small, cross functional group that plans, commits, implements, demos, and continuously improves together
- Self-organized, self-managed



Scaling Agile to large systems or organizations

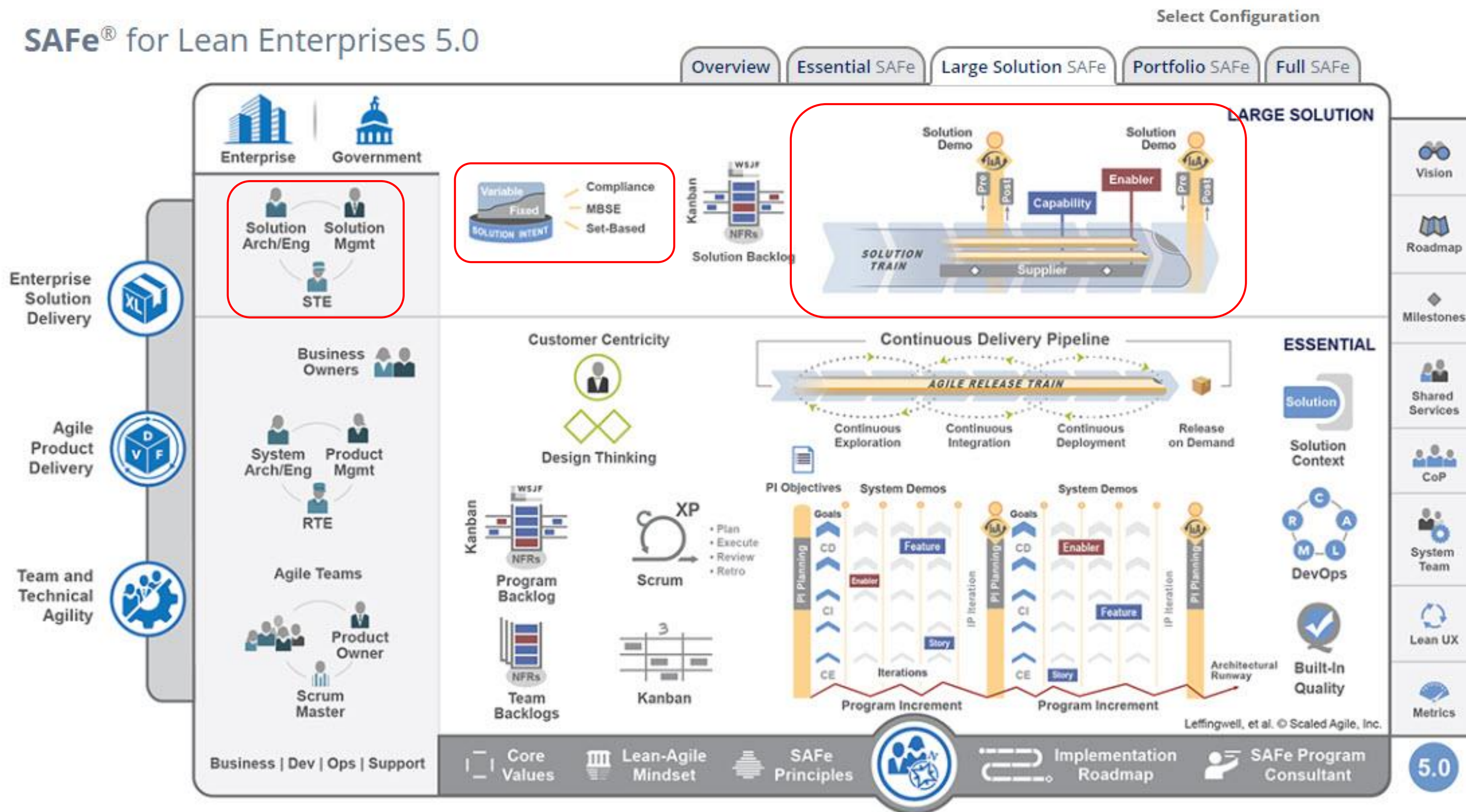
Information

Roles Backlog Cadence Alignment events



Scaling Agile to (very) large systems and enterprises

Information



A man wearing a grey baseball cap and a black t-shirt is seen from behind, looking at a wall covered with various documents, charts, and posters. The wall appears to be a workspace or a workshop. There are orange labels for 'Team B' and 'Team C'. A prominent yellow poster with a stylized 'QA' logo is visible. Other documents include a rocket illustration, a bar chart, and a large handwritten 'D'.

Know the characteristics of a specification workshop

Specification Workshops create alignment and shared understanding

- Collaboratively discover and communicate solution's specification and prioritization
 - Create and maintain system's single source of truth
 - Socialize ideas from multiple experts
 - Perform engineering work continuously
 - Allow architectures to be defined incrementally
 - And designs to emerge
 - Record decisions in specifications

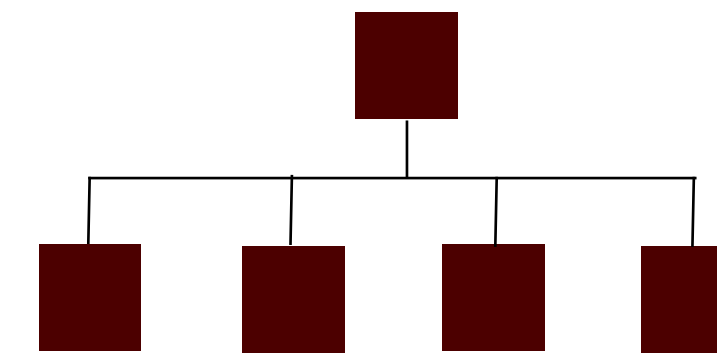
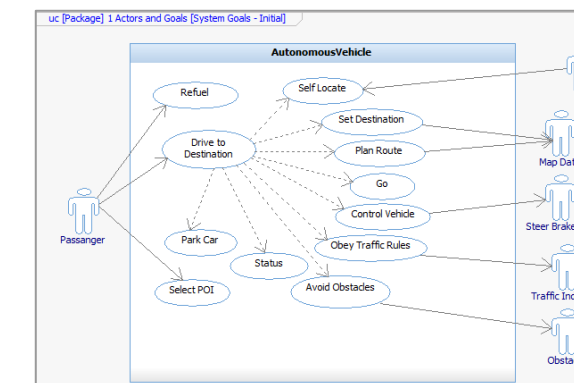
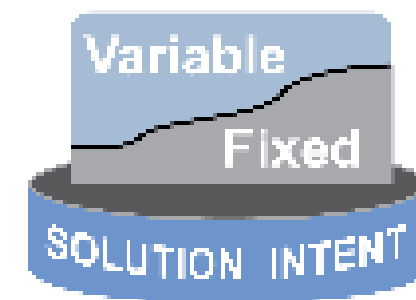


Use SE Workshops to create specifications and plans

- Workshops elaborate the architecture and design over time
- Keep options open, elaborate detail at appropriate time
- Use MBSE to identify and model coarse-level system behavior and structure
- Define plan to realize solution

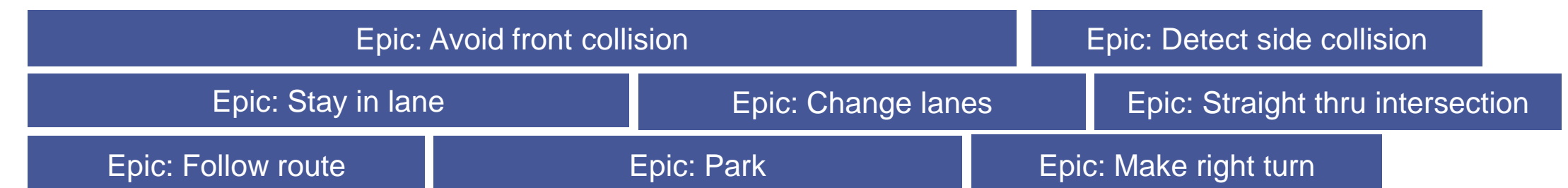
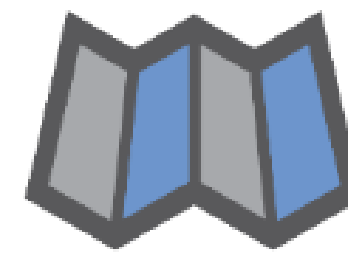
System structure and behavior

- Create coarse understanding of behavior, architecture
- Sets solution and technical vision, creates alignment



Development roadmap

- Plan to incrementally realize solution
- Focus teams on near term goals, learning



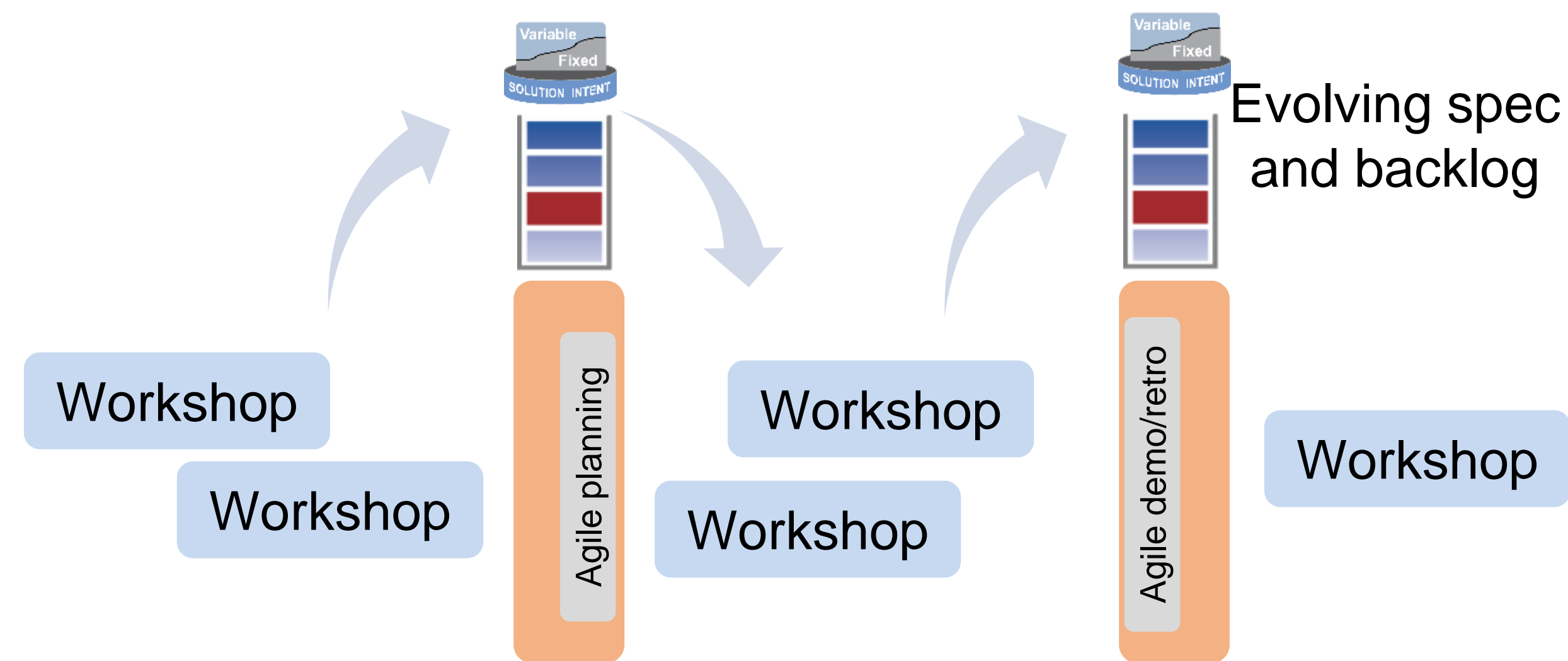
Program Backlog

- Incrementally define backlog items
- Features, Enablers, and NFRs

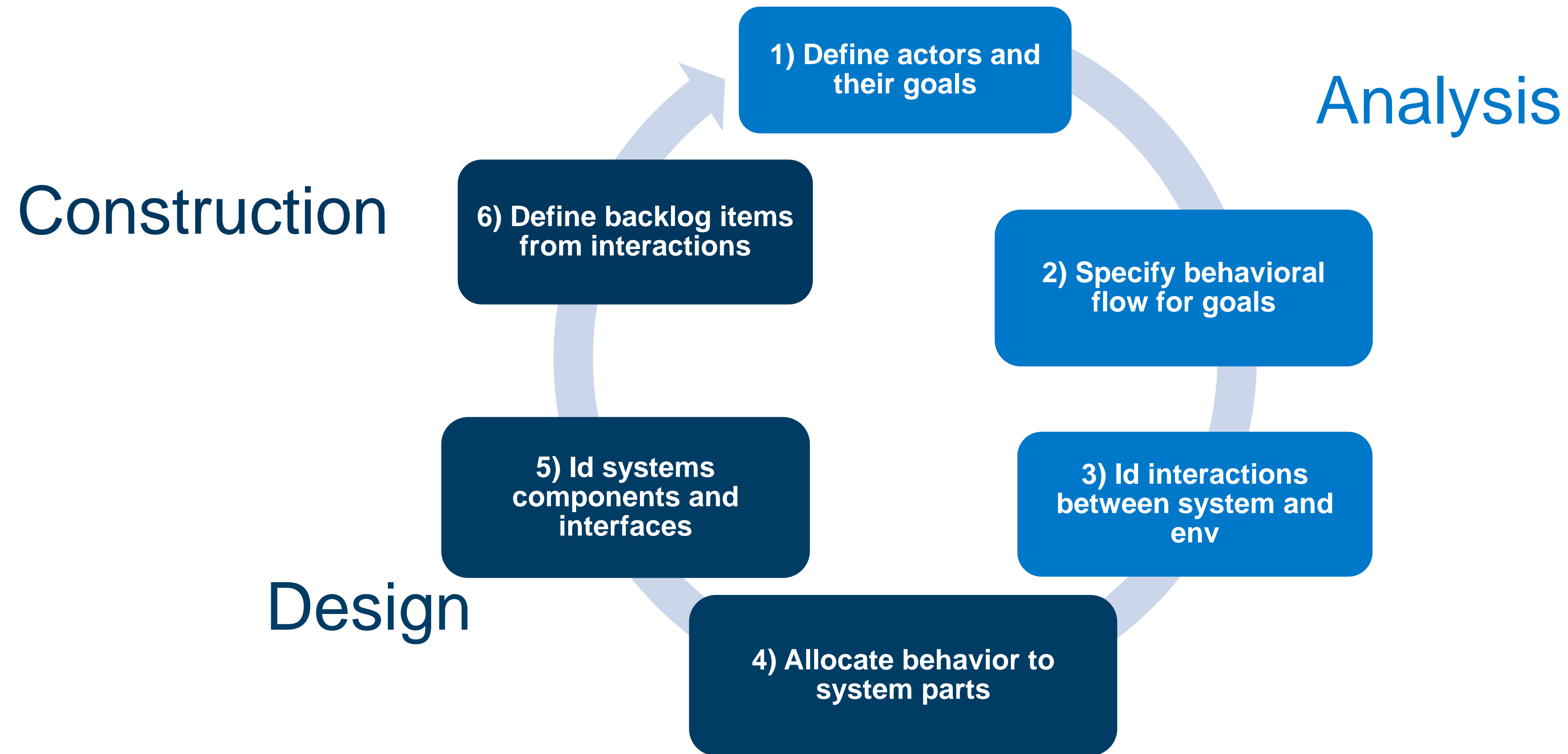


Make workshops part of cadence-based feedback

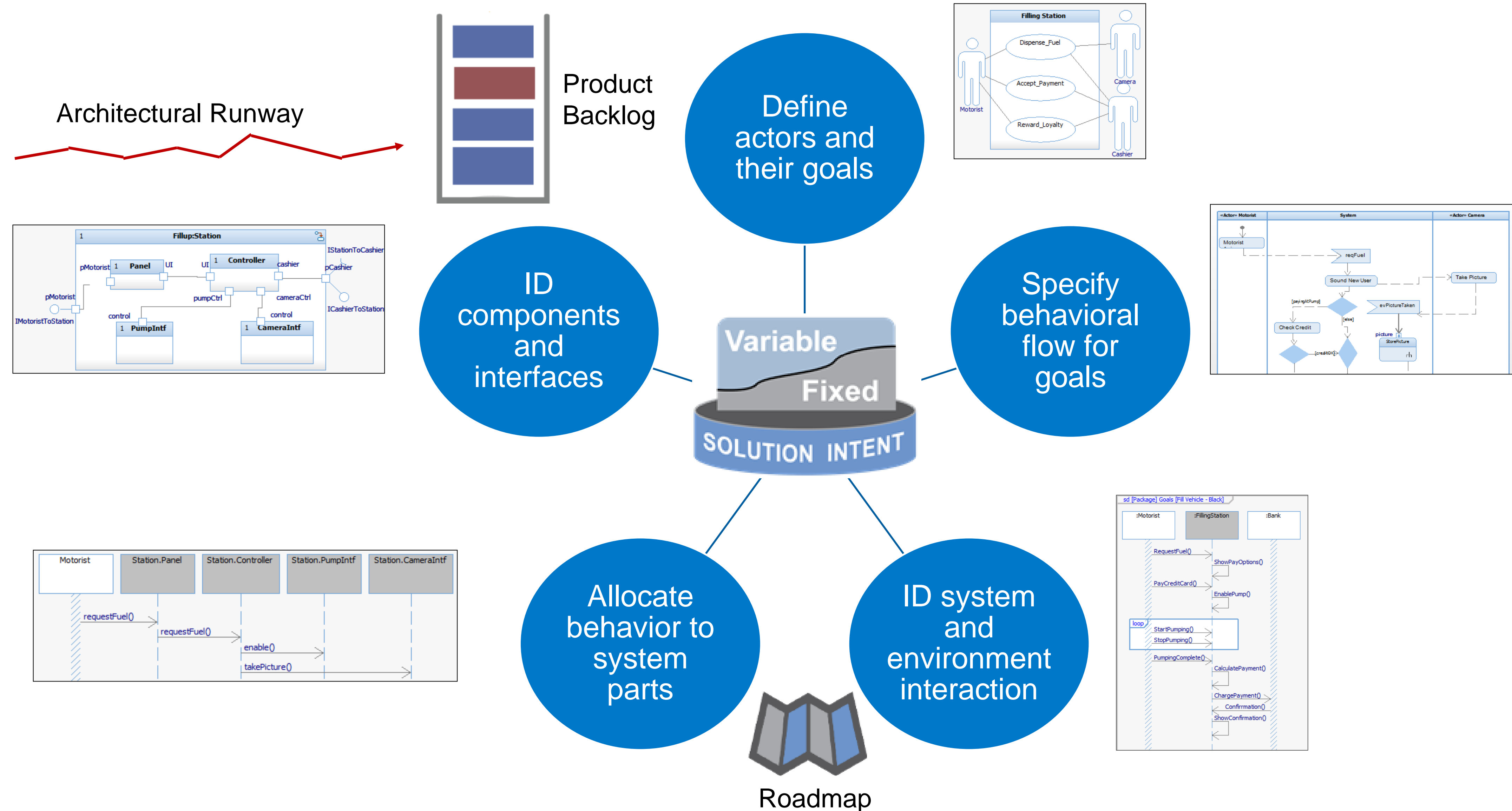
- Workshops provide focus on impending system needs – what to build and/or explore in near-term increments
- Regular cadence provides fast feedback on decisions
- Results in a more complete, consistent technical architecture
- *Workshops are in addition to regular agile and/or SAFe ceremonies*




Iteratively develop specifications



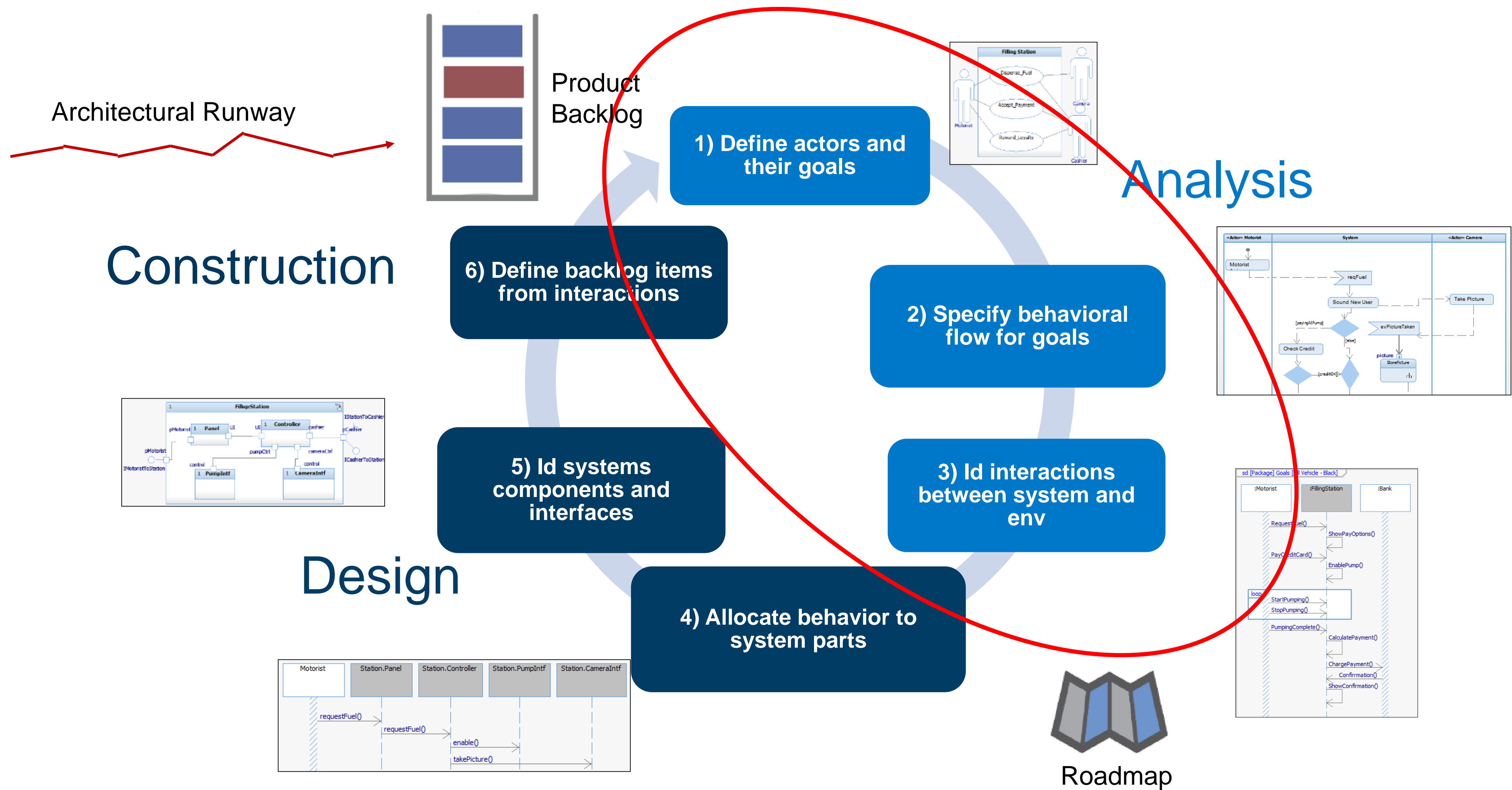
Use SysML for system specifications



A detailed wireframe model of a classic convertible car, shown from a side profile. The model is composed of a dense network of white lines on a black background, representing the car's body, wheels, and interior. The front of the car features two circular headlights with red mesh-like interiors. A blue horizontal bar is overlaid across the middle of the image, containing the text "Let's look at the Analysis Phase".

Let's look at the
Analysis Phase

Iteratively develop specifications



Analysis – Step 1

Define Actors and their Goals

Stakeholder Needs– Autonomous Vehicle Controller



- Driver sets destination and says “go”
- Driver may change or add to destination
- Driver may select to “refuel” in which case the route will be suspended and vehicle will drive to nearest fuel location
- Vehicle shows real-time route and destination status
- Vehicle should notify driver when refuel necessary and should advise when distance to station is a concern
- Driver can select interests to be notified – restaurants, museums, historical land marks, etc.
- Driver can cancel automation at any time

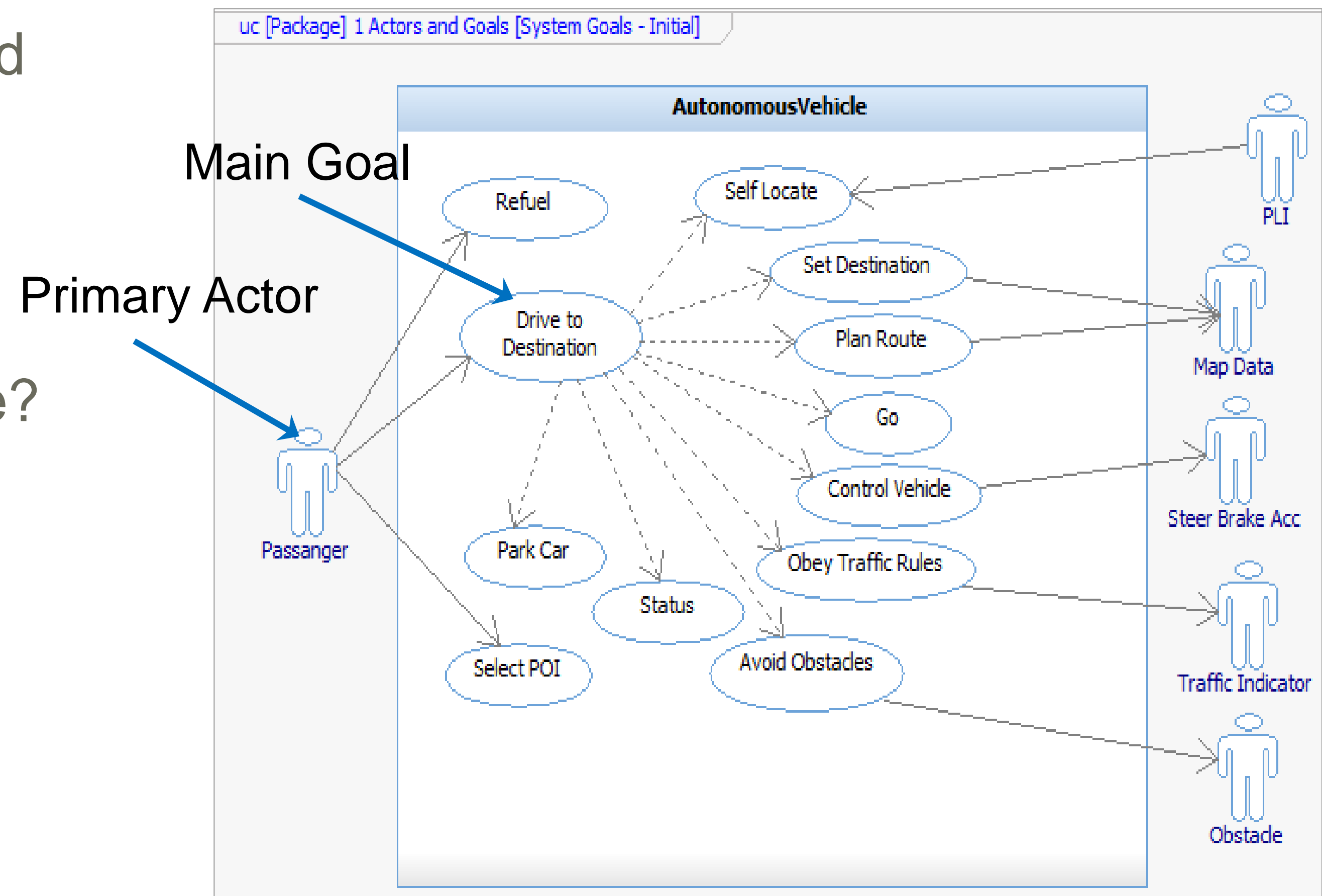
Non-Functional



- Obey all traffic laws
- Do not kill anyone
- Do not get in an accident
- Park in the shade during the summer

1) Apply Use Case modeling – actors and their goals

- ✓ Start with 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?
- ✓ With which actors do they interface?

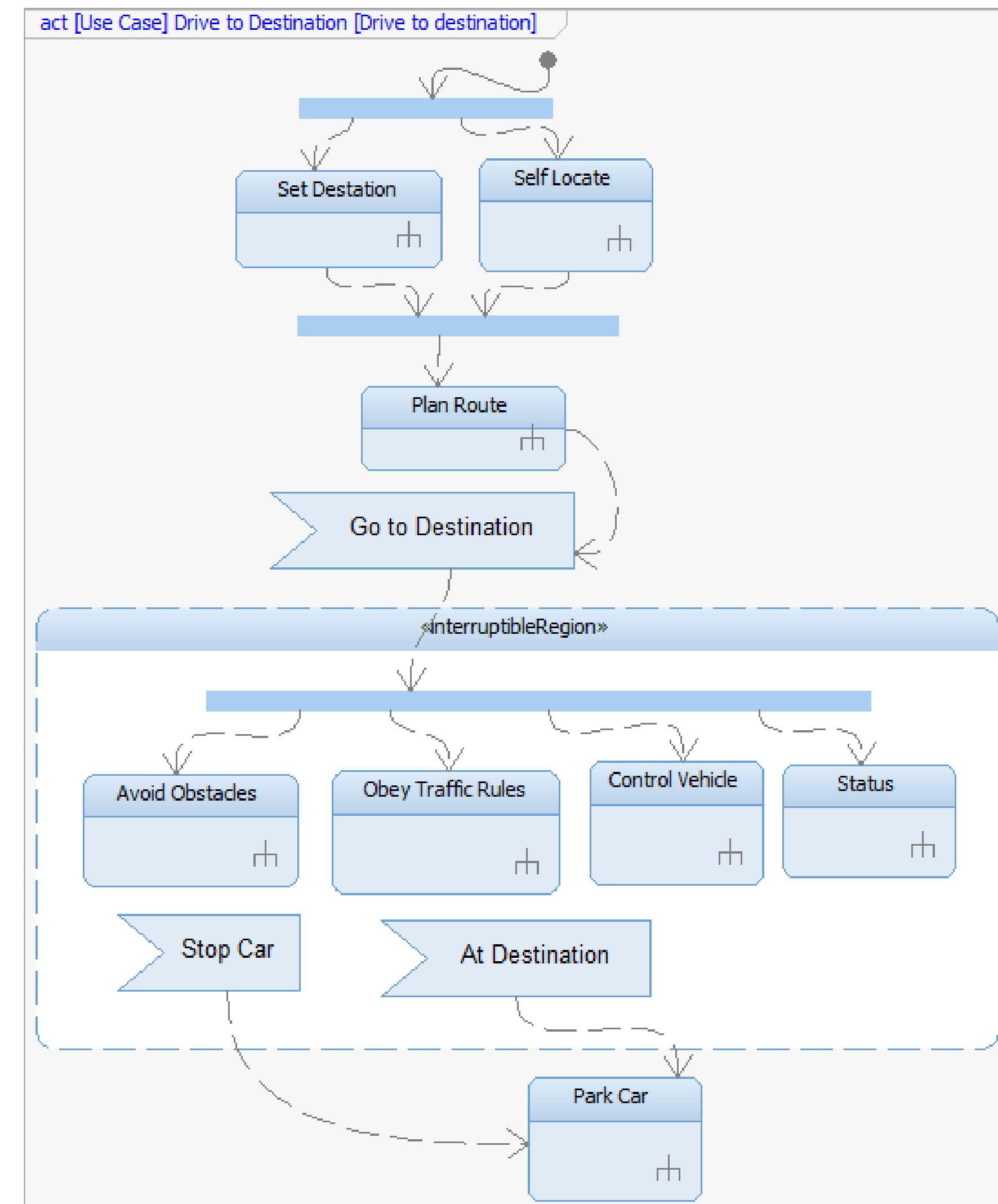
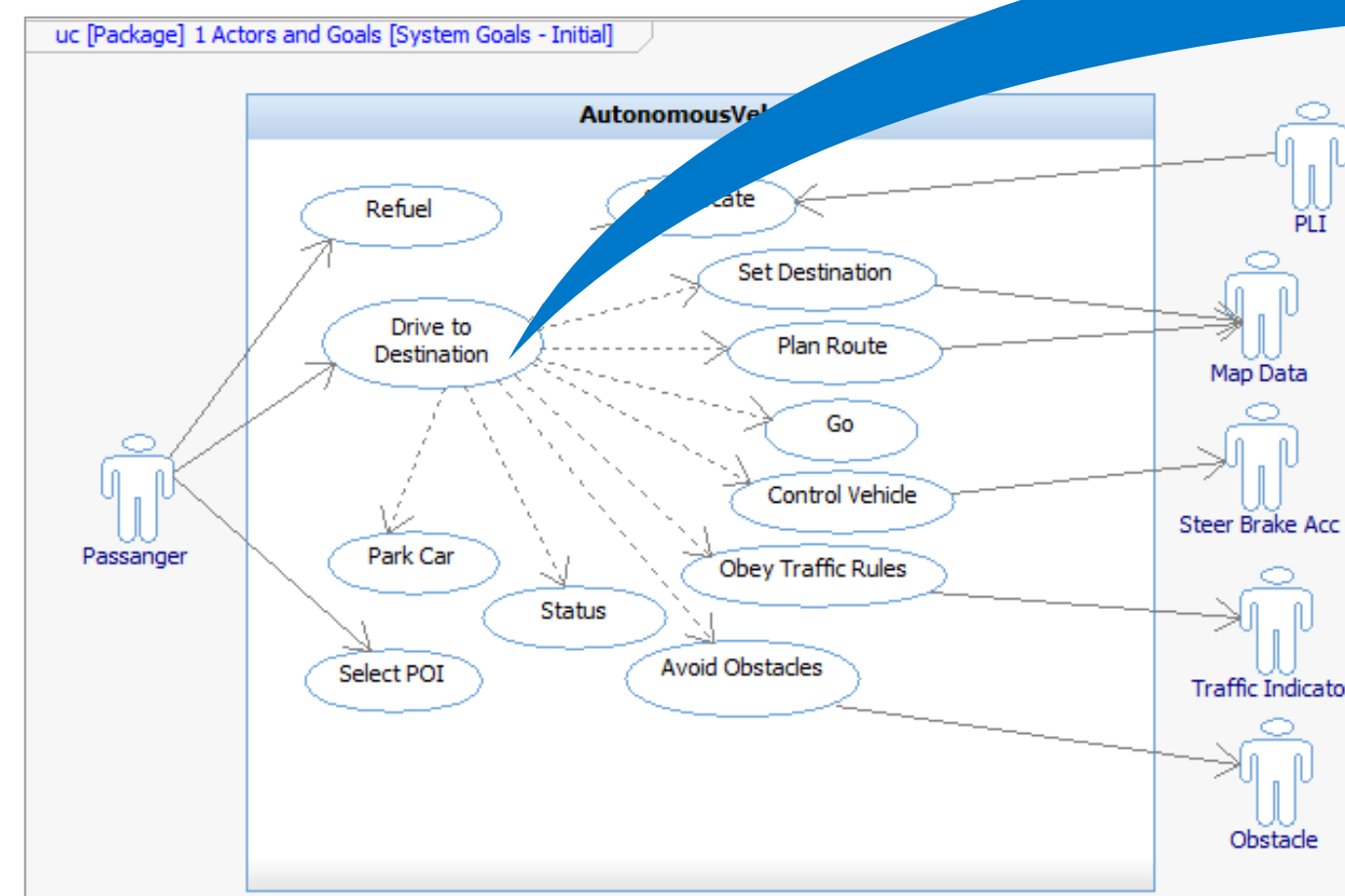


Analysis – Step 2

Specify Behavioral Flow for Goals

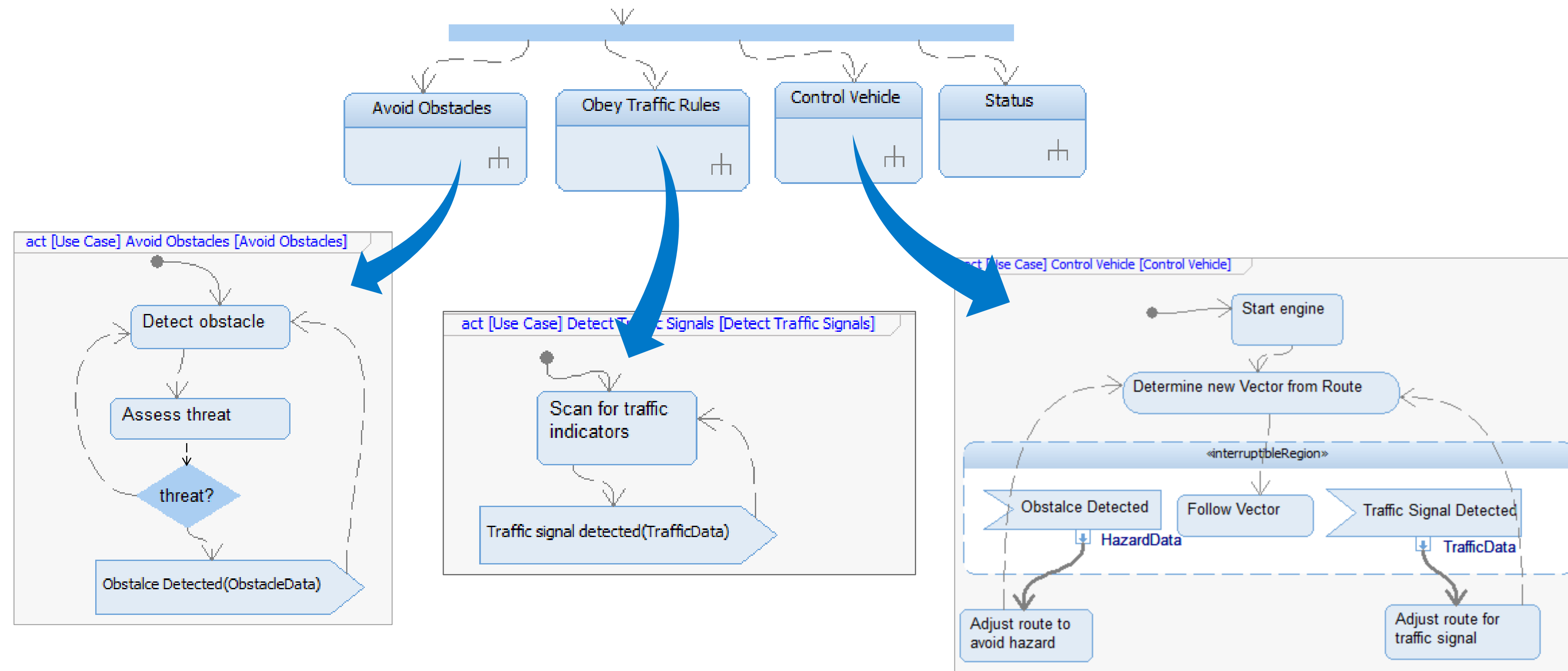
2a) Specify behavioral flow for goals

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



2b) Specify behavioral flow for sub-goals

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

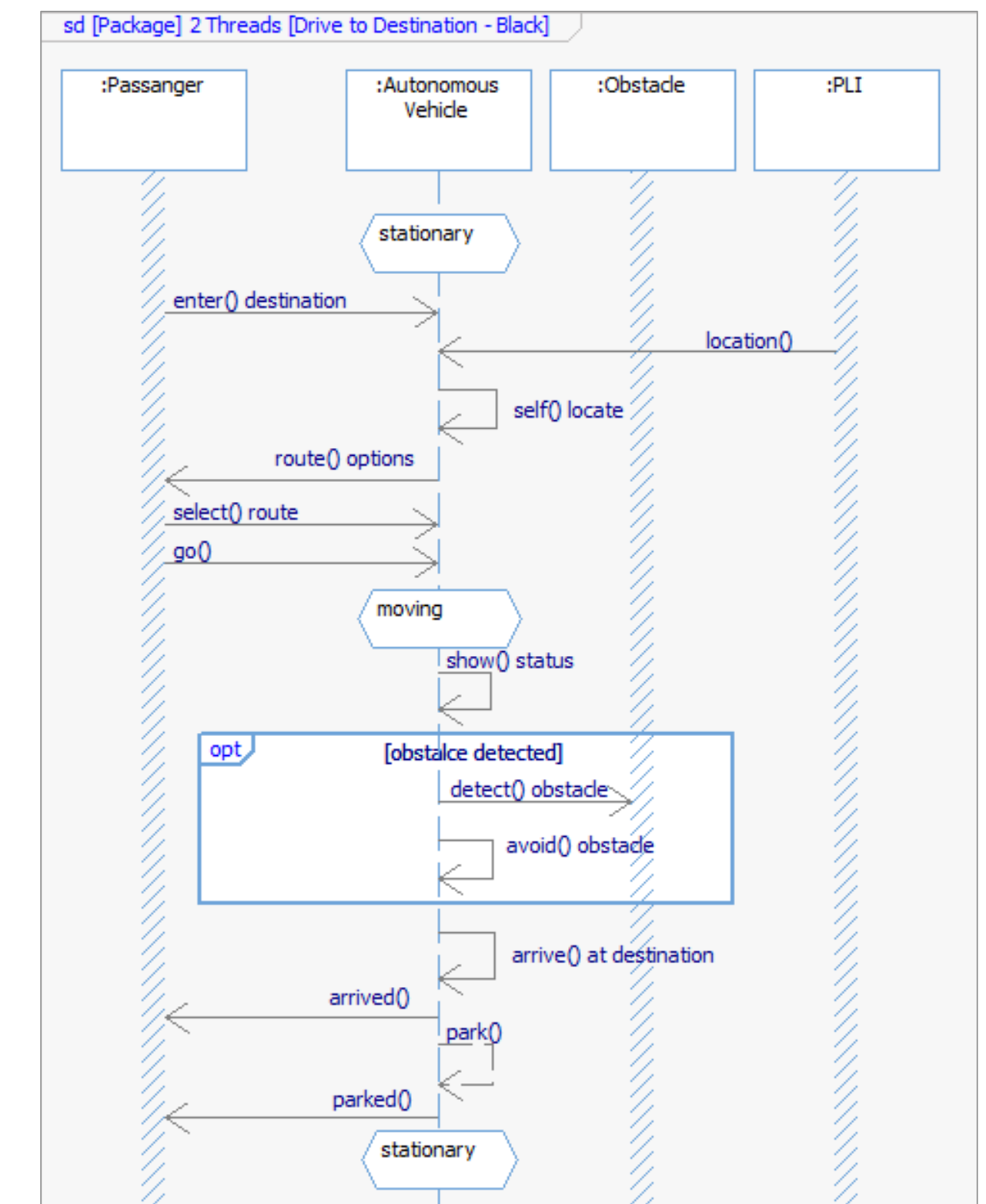
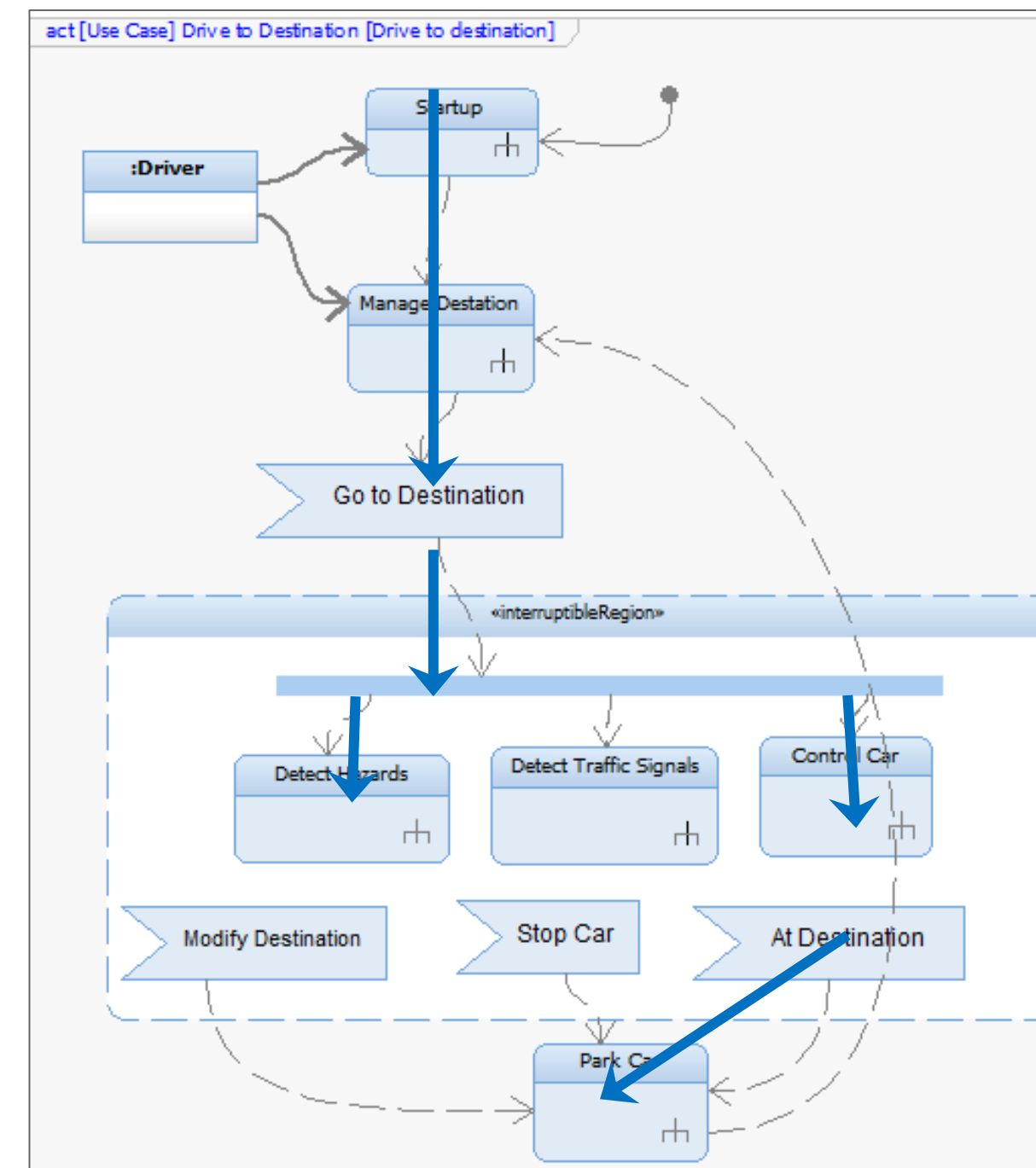


Analysis – Step 3

Identify Interactions between System and Environment

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
- ✓ “Highly visible” system responsibilities are shown as self-interactions
- ✓ System is a black box, no internal components
- ✓ Useful for system-level tests



Post Analysis Identify Epics and Build The Epic Roadmap

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

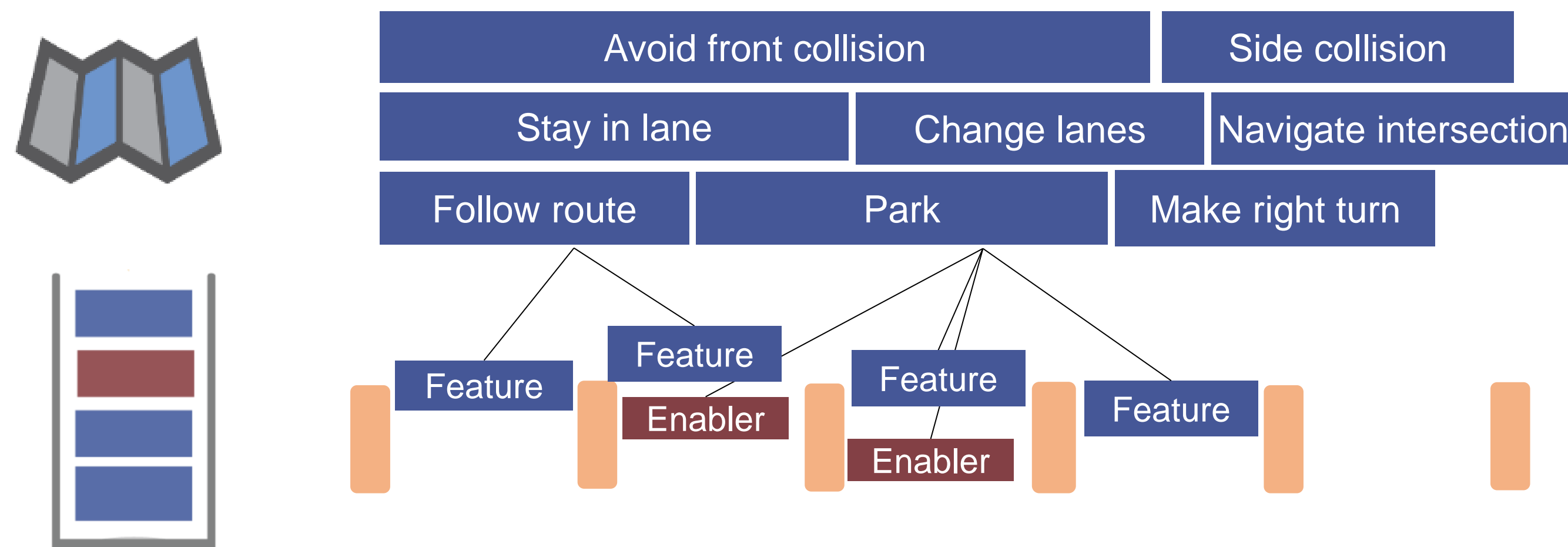
Epic: Obey Traffic Lights

Epic: Simple drive to destination

Epic: Detect forward vehicle

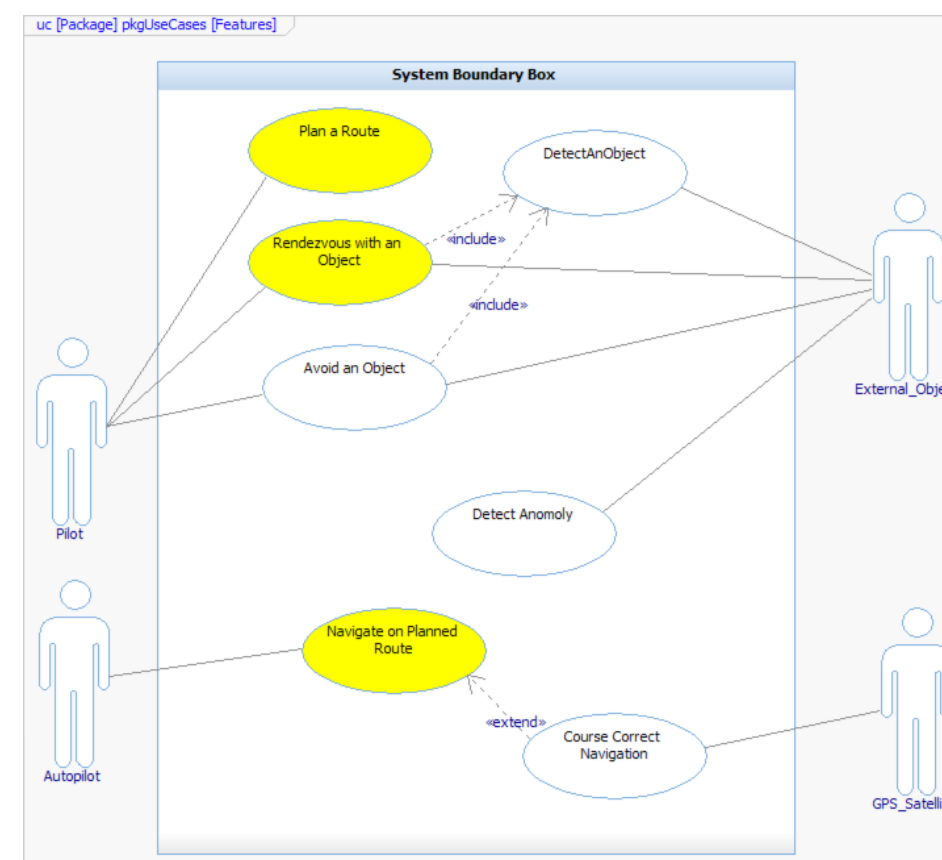
Roadmap forecasts Epics over life of program

- ✓ Similar to planning packages that will be decomposed into time-boxed Features or Stories (described in next section)
- ✓ Order the Epics using either an MVP or WJSF approach to build the roadmap
- ✓ Use cadence to focus teams and provide feedback on progress and feasibility



Expand knowledge and decisions over time; manage change

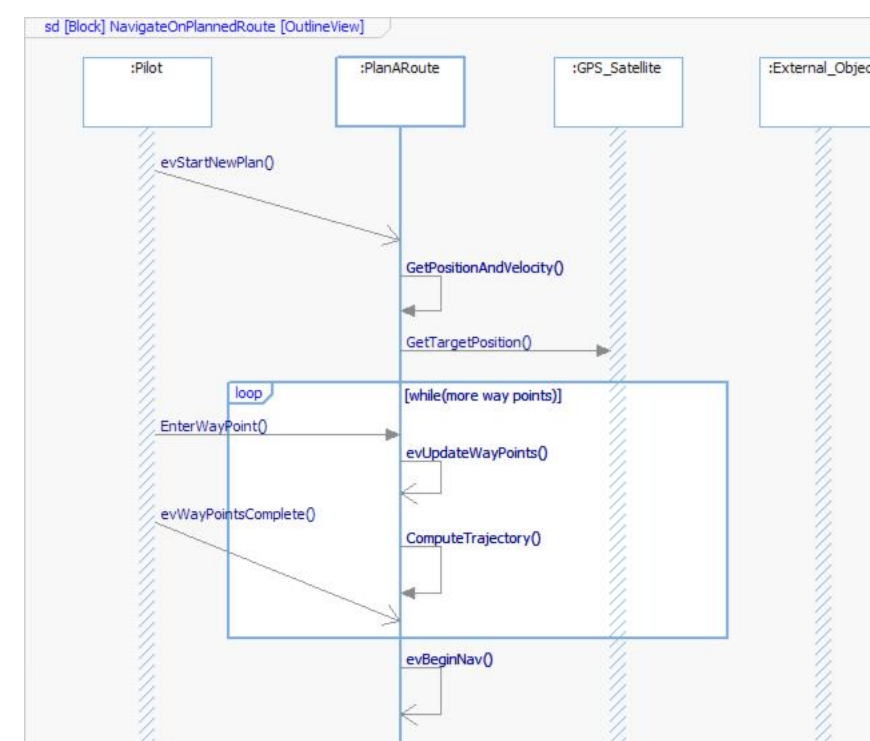
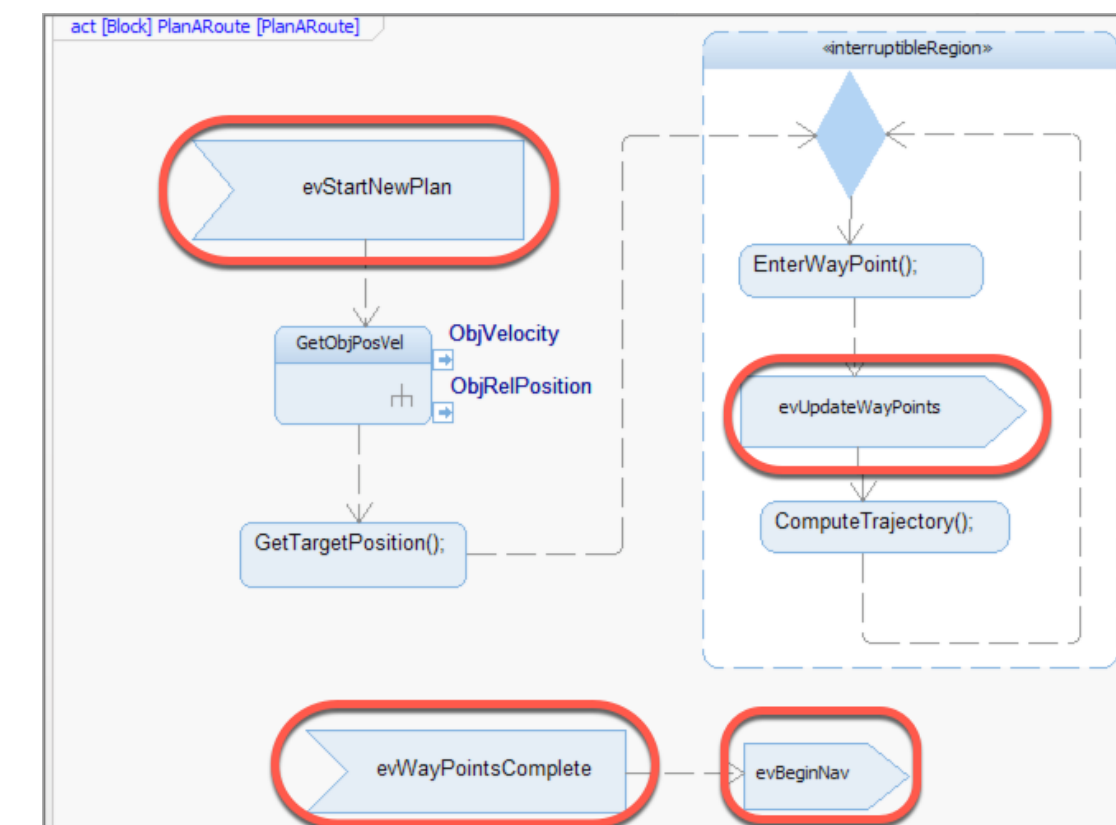
- Discover and explore other actors and their goals
- Elaborate additional scenarios (exception, alternate, rainy)
- Each scenario creates new threads/Epics on the backlog



**Define actors
and their goals
(Alpha first)**

**Specify
behavioral flow
for goals**

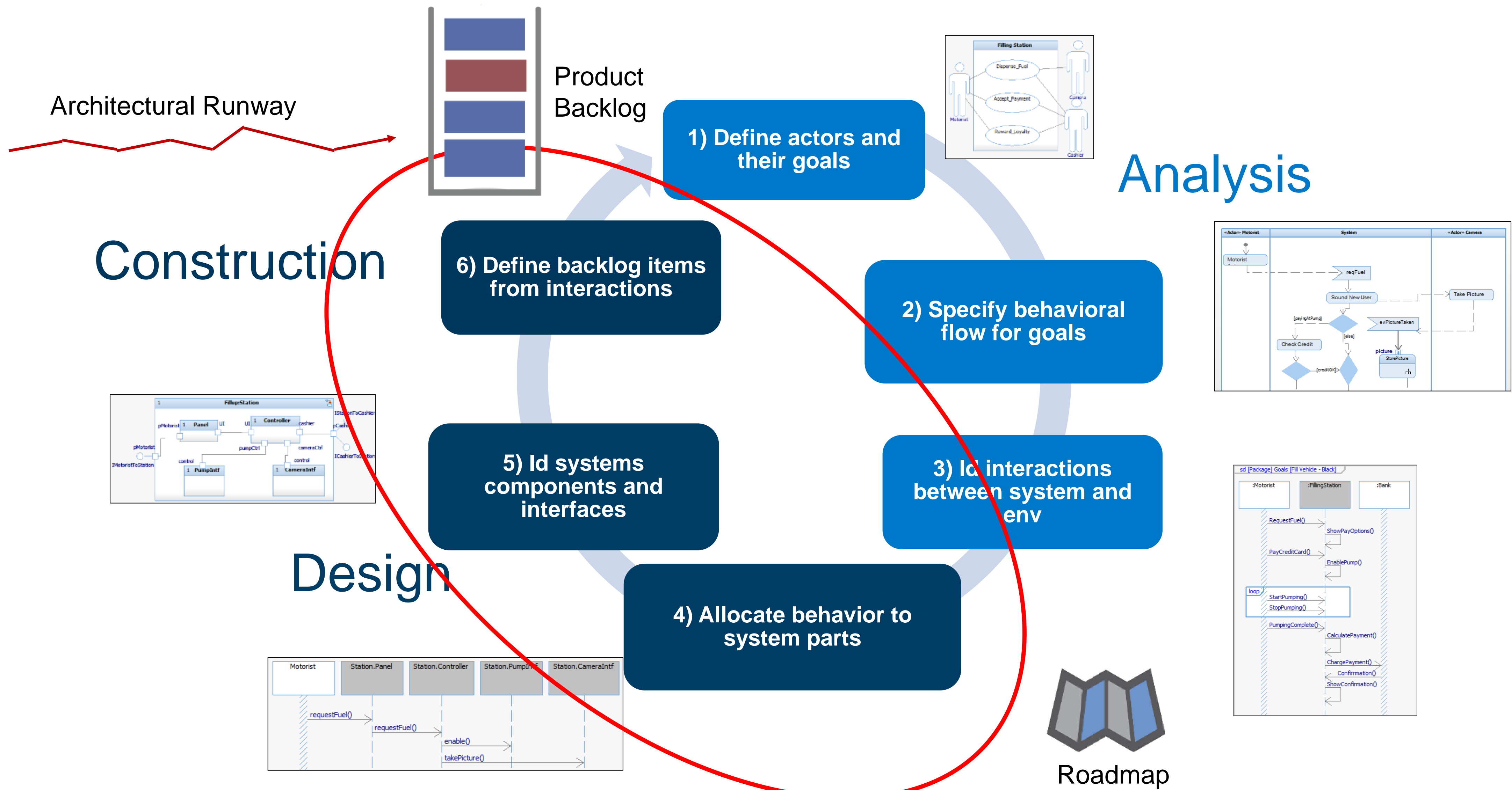
**Id stimulus
between system
and env**



A man wearing a grey baseball cap and a black t-shirt is seen from behind, looking at a wall covered in various project planning documents. The wall features several large sheets of paper with text, tables, and charts. Two orange sticky notes are visible, labeled 'Team B' and 'Team C'. A prominent yellow sticky note with a stylized 'QA' logo is in the center. In the bottom left corner, there is a small red and white rocket icon. A large blue banner with white text is overlaid on the right side of the image.

Prior to each Program Increment Planning Session
go through the Design and Construction Phase

Iteratively develop specifications

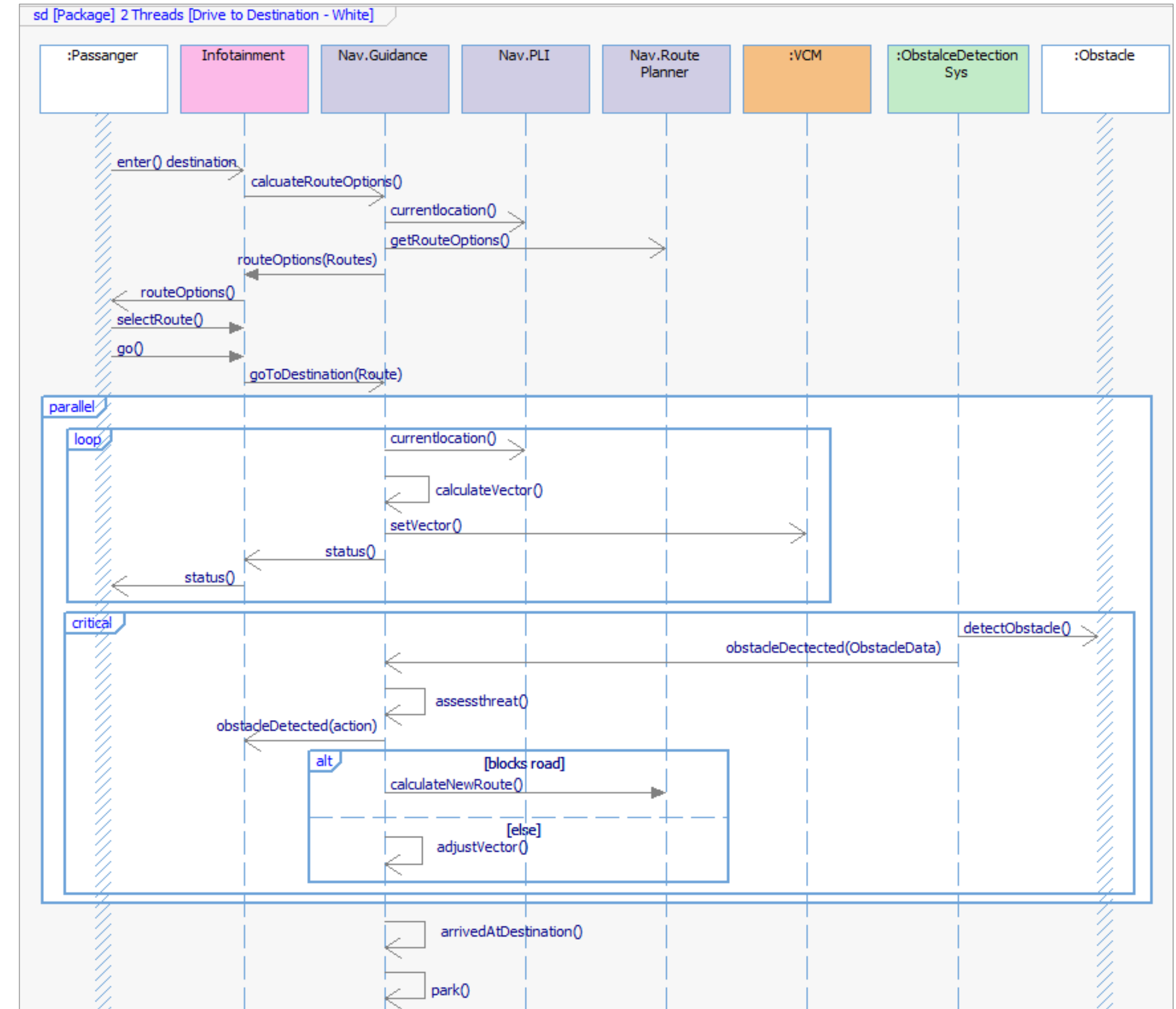


Design – Step 4

Allocate Behavior to System Parts

4) Allocate behavior to system parts

- For each external interaction, decide how system elements realize the behavior
 - What system part receives the interaction and what is the system's response?
 - What system part sends the interaction and what led to system sending it?
- First step in design - determine how behavior is realized by system elements
- Discover system parts and allocate responsibilities to them



Design – Step 5

Identify System Components And their Interfaces

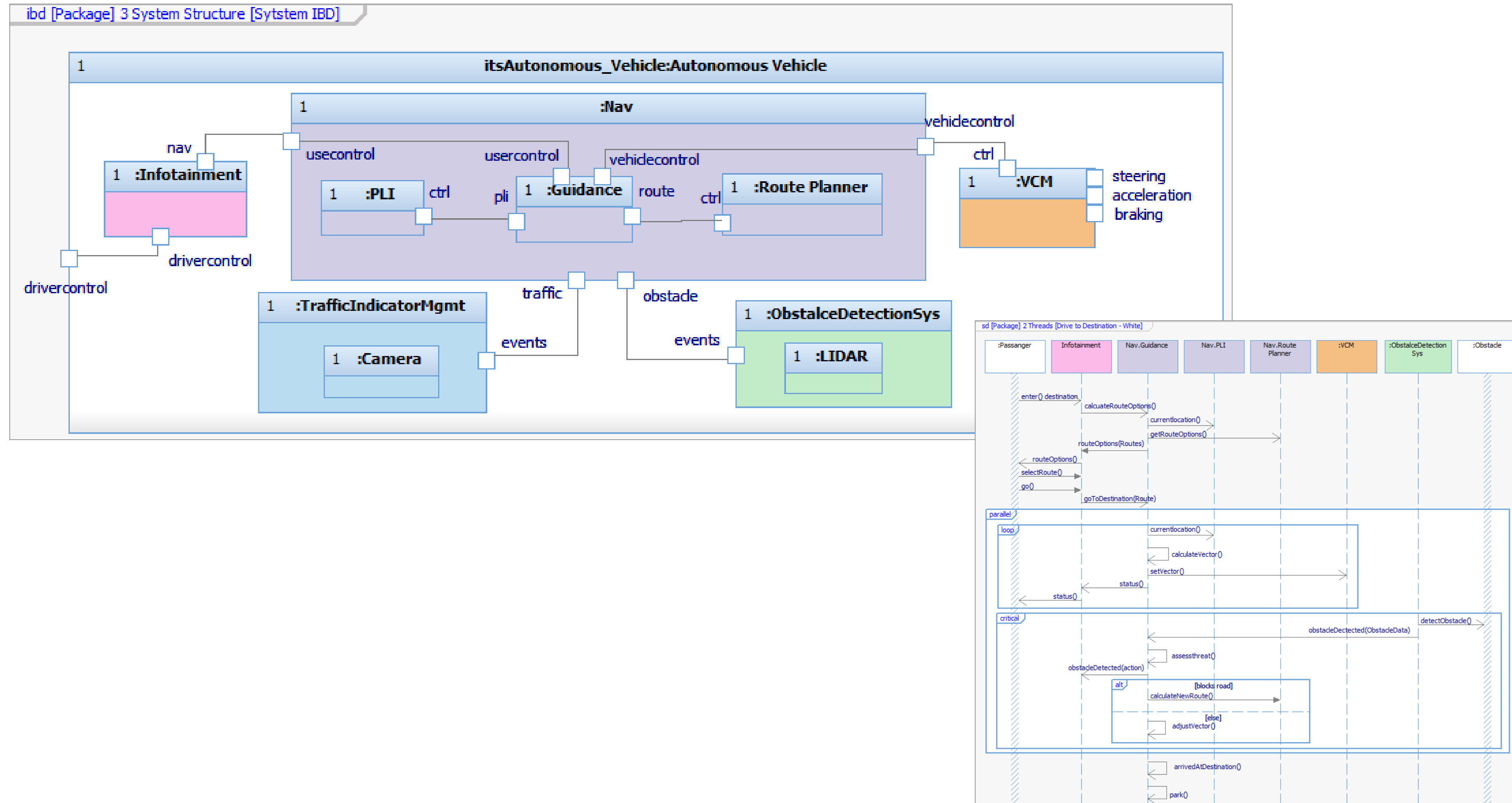
5) Define system components and interfaces

- Blocks decompose system's hierarchical structure
- Ports define interactions into and out of a block
- Interactions that must be consumed or provided
- Interfaces and behavior serve as requirements for block implementation (ICD)
- Decomposition scales to extremely large systems

The purpose this exercise is not to build the ultimate block diagram with interface blocks and the like, but to decompose the SUD into its hierarchical structure. As well, the interfaces or connections among the blocks need to be identified.

5) Define system components and interfaces

- Example includes internal components for Nav (optional, shown as example)



Construction – Step 6

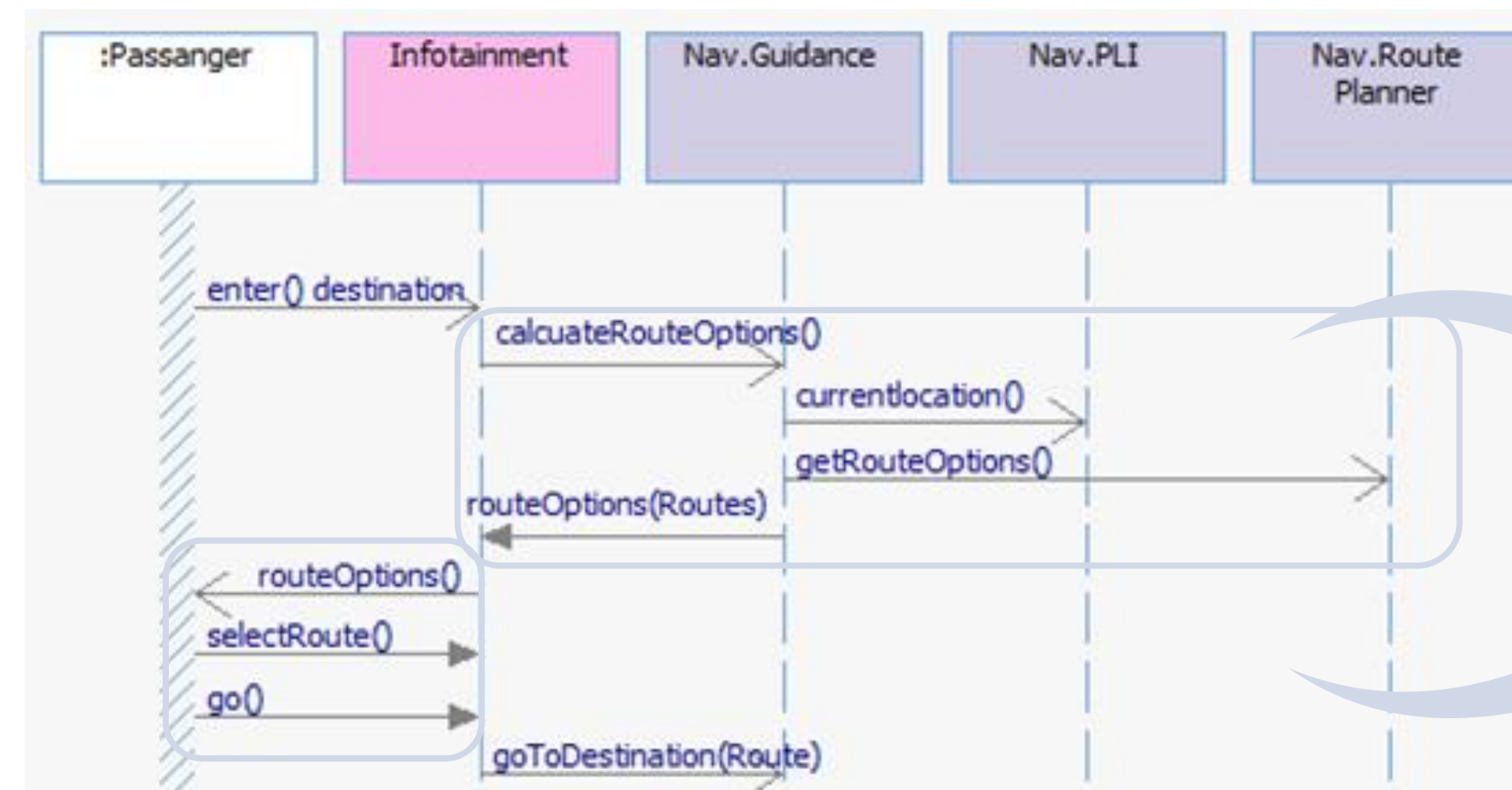
Define Backlog Items from Interactions

6) Define backlog items from interactions

- 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



Feature
Enabler
Feature
Feature
Enabler

Manage non-functional requirements (NFRs)

- NFRs constrain backlog items' implementation; NFRs are not backlog items themselves
- Workshops will uncover both behaviors and constraints
- Record NFRs as they are found – trace for compliance later
- Enablers may be required to support features adherence to NFRs

NFRs

- Obey all traffic laws
- Don't kill anyone
- Don't get in an accident
- Park in shade during summer

Epic: Drive to Destination

Epic: Obey Traffic Lights

Epic: Detect forward vehicle

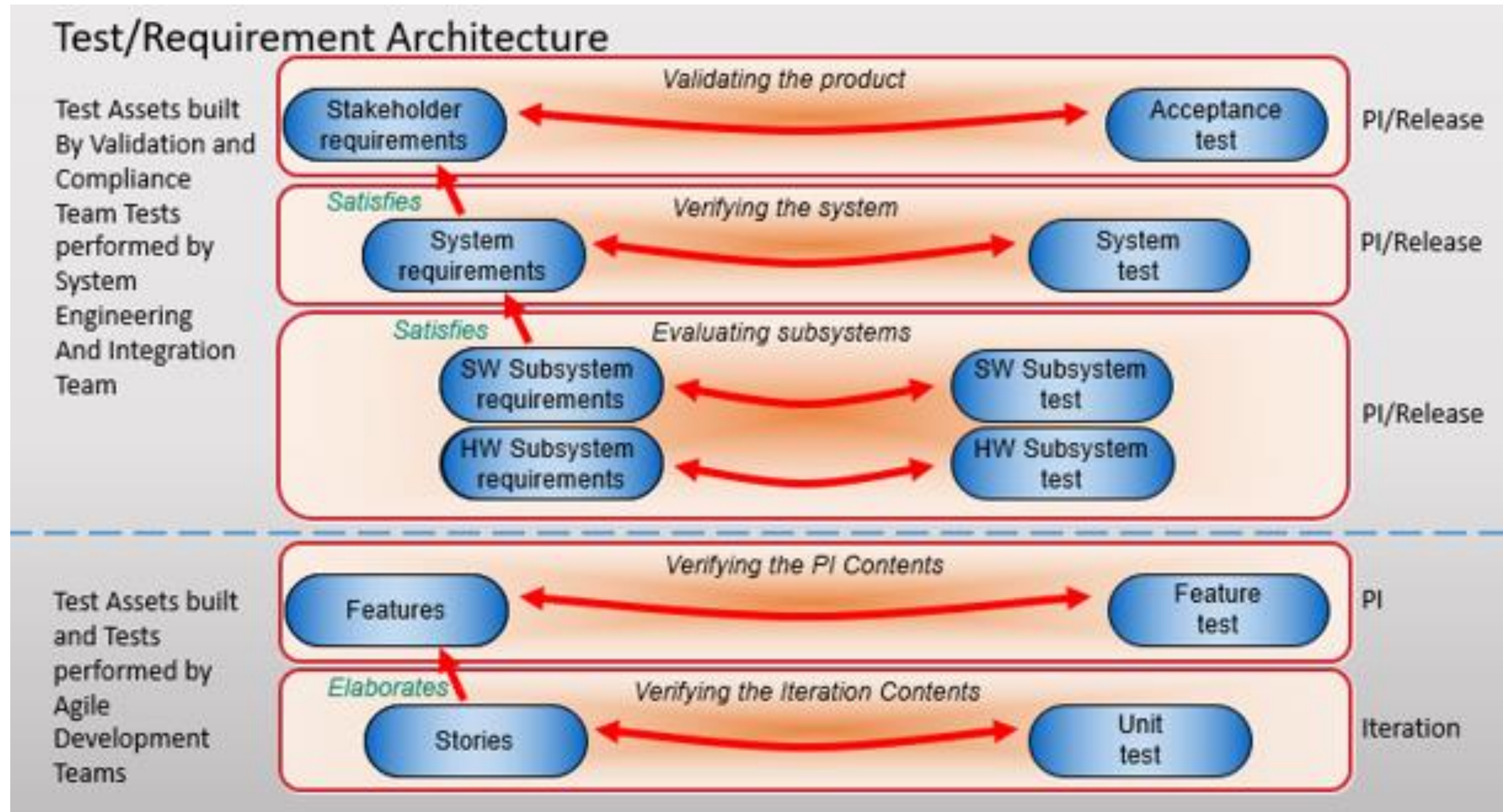
Epic: Detect side obstacles

Epic: Stay in lane



Putting it all together

Requirements & Testing in a Regulated Environment within SAFe



Putting it all together

Inputs

Behavioral Analysis

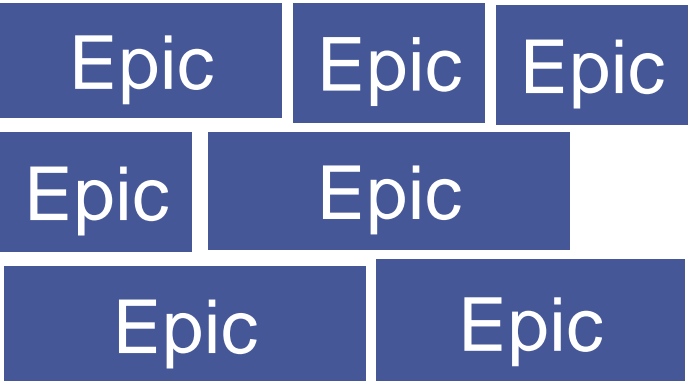
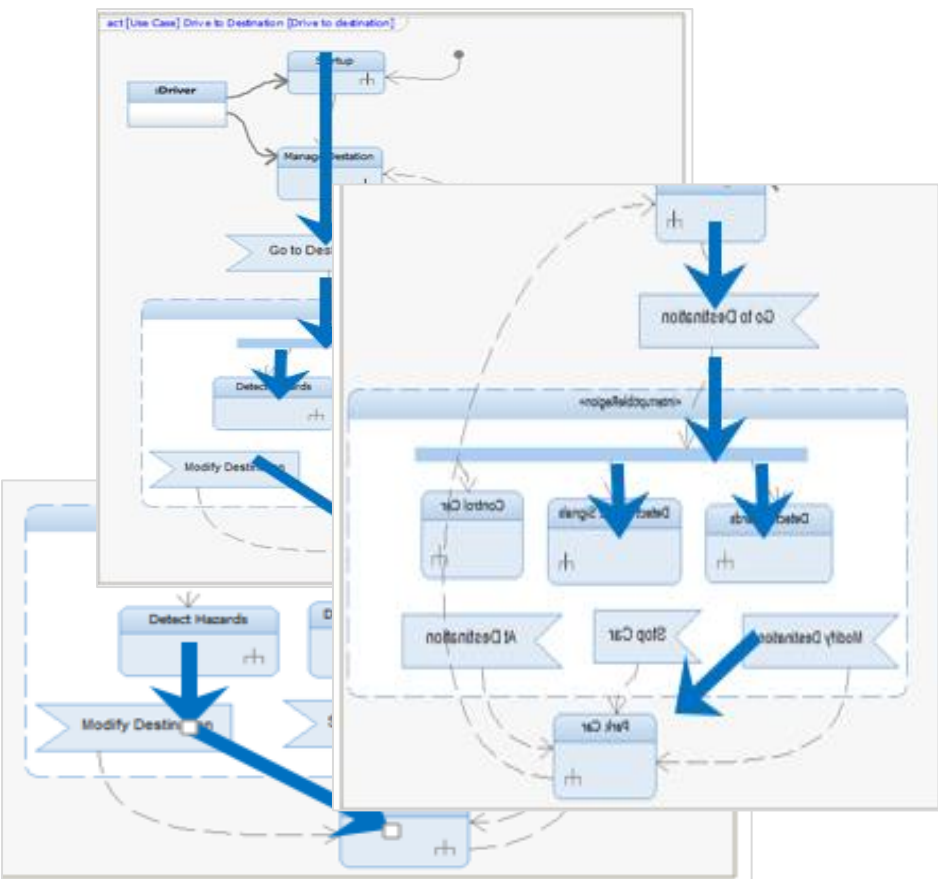
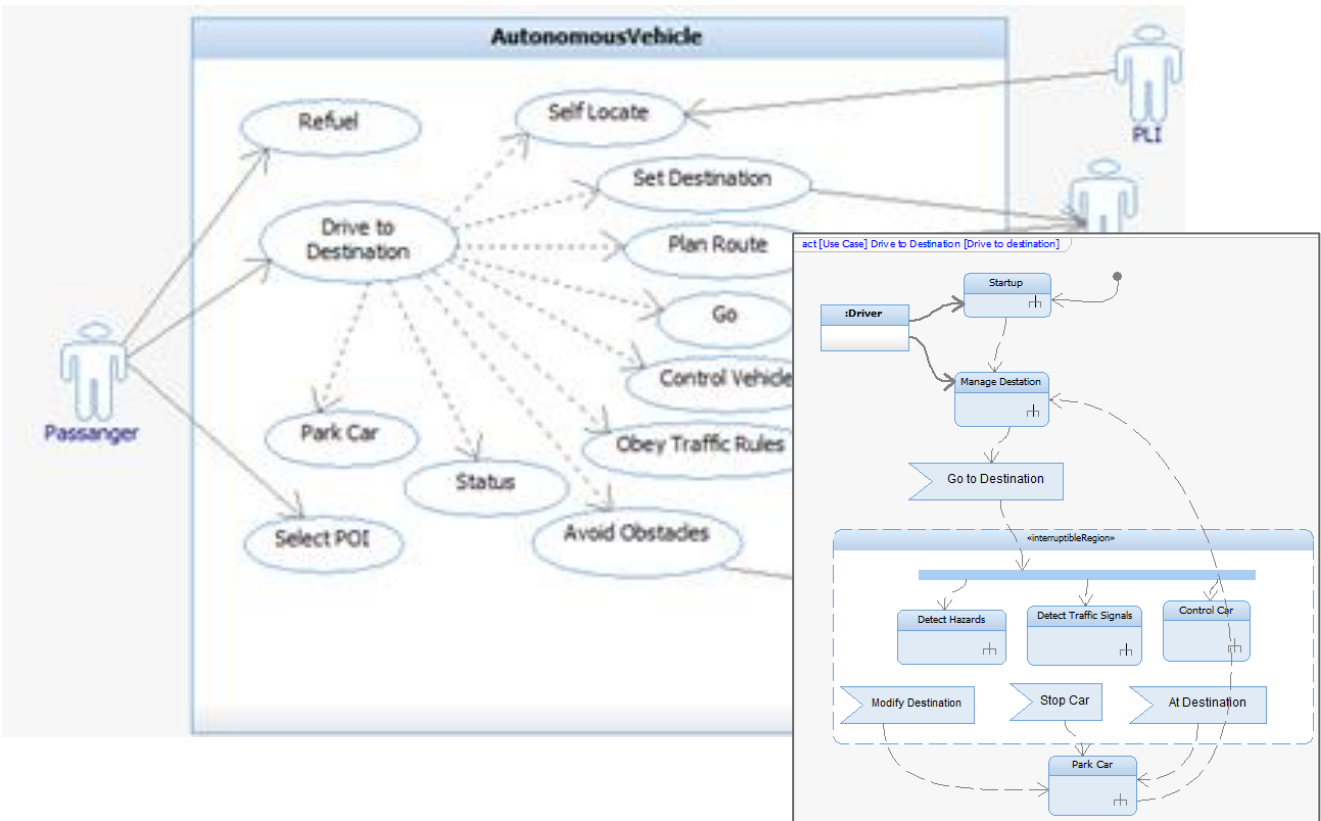
Roadmap
(Planning Packages)

Proposal

Stakeholder Needs

Proposal

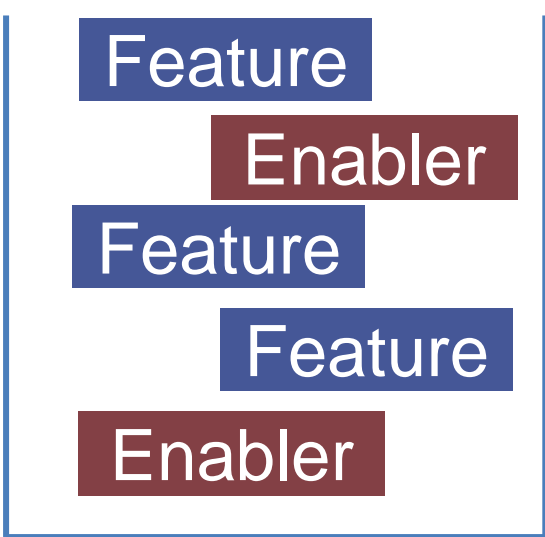
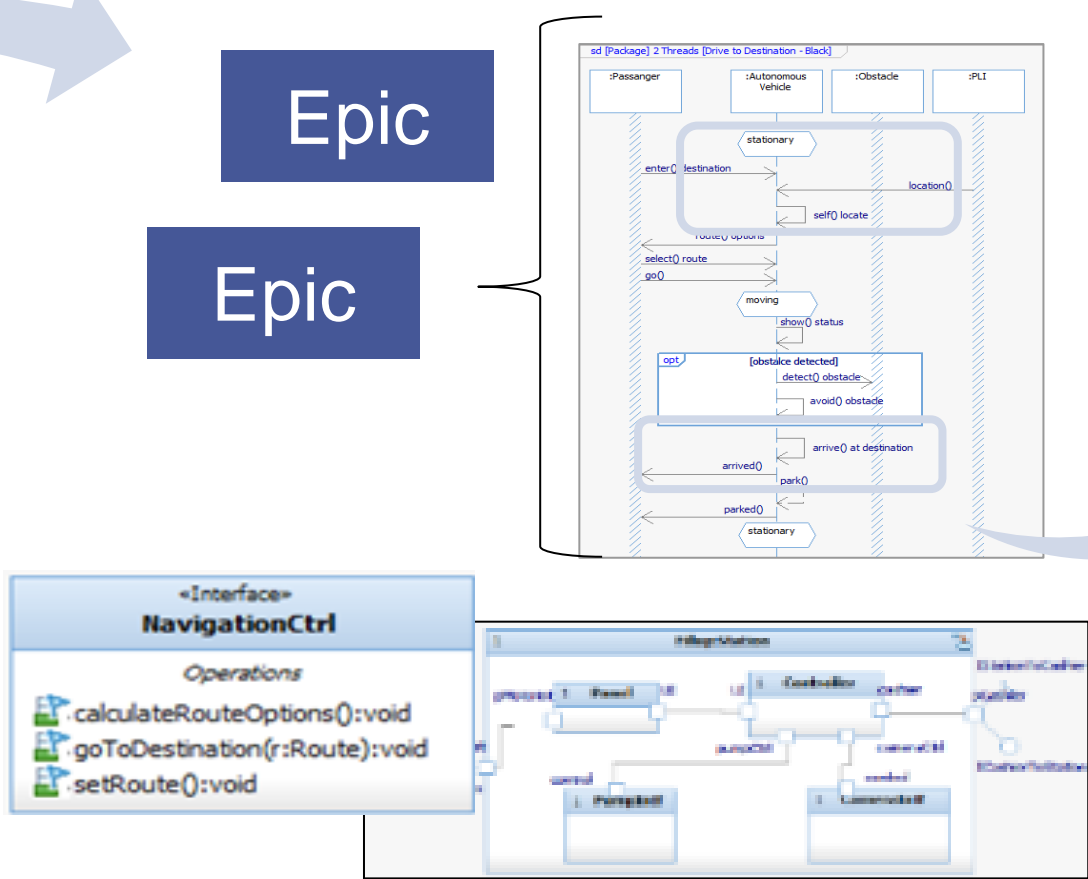
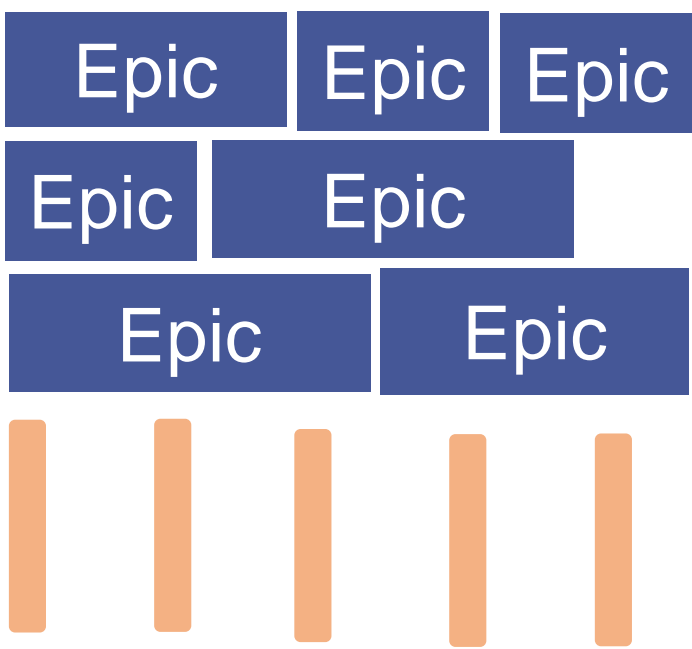
Previous sys spec



Analysis/Design

Implementing

Program Execution



Modeling role guidance

If solution is composed of one ART, what roles should be involved in the following activities?

- Create Use Case and Activity diagrams
- Create black box and white box sequence diagrams,
- Build the Epic Roadmap
- Identify the Features for PI
- Work on a daily bases on Activity diagrams, and with teams on sequence and block diagrams

How should System Engineers be incorporated into an ART?

Group discussion

- Single System Engineering Team
- Embedded with each Agile Team
- Hybrid model

Summarize Specification Workshop Value

- What is this process?
 - Collaboratively build a solution through emergent specifications and an agile roadmap
- What value does it provide?
 - Get what you want, not what you specified
 - Continuously align everyone on delivering greatest value frequently and reliably
 - Explore alternatives to make best economic decisions
 - Adapt to new information, both external and internal to the program
- What is our ask of customer/business?
 - Participation with us in regular agile ceremonies for alignment, review, and adapting
 - Appreciate lean principles of cadence, synchronization, flow, WIP, adapting to new knowledge, emergent design, and taking an economic view, and apply them to traditional EVM, Schedule, CDRLs, etc.

Contact us:

www.321gang.com

Peter@321gang.com

Questions?

Additional Resources

- ✓ Getting Started with the Scaled Agile Framework (SAFe): bit.ly/321Gang-SAFe
- ✓ SAFe & Model-Based Systems Engineering: bit.ly/321Gang_MBSE_SAFe
- ✓ IT Revolution White Paper: Industrial DevOps (Contributors: NGC & LMCO)
- ✓ F-22 SPO Advanced Tactical Fighter & SAFe: bit.ly/F22-SAFe
- ✓ Don Reinertsen - The Logic of Flow: bit.ly/ReinertsenLogic-of-Flow
- ✓ AUTONOMY. MASTERY. PURPOSE: bit.ly/Autonomy_Mastery_Purpose