When worlds collide: Hardware Engineering meet Lean-Agile development

Harry Koehnemann

SAFe Fellow and Principal Consultant

Scaled Agile, Inc.

SCALED AGILE®

Agenda

- The value of continuous learning
- Explain Agile product development
- Specify the system incrementally
- Design for change
- Frequently integrate the end-to-end solution

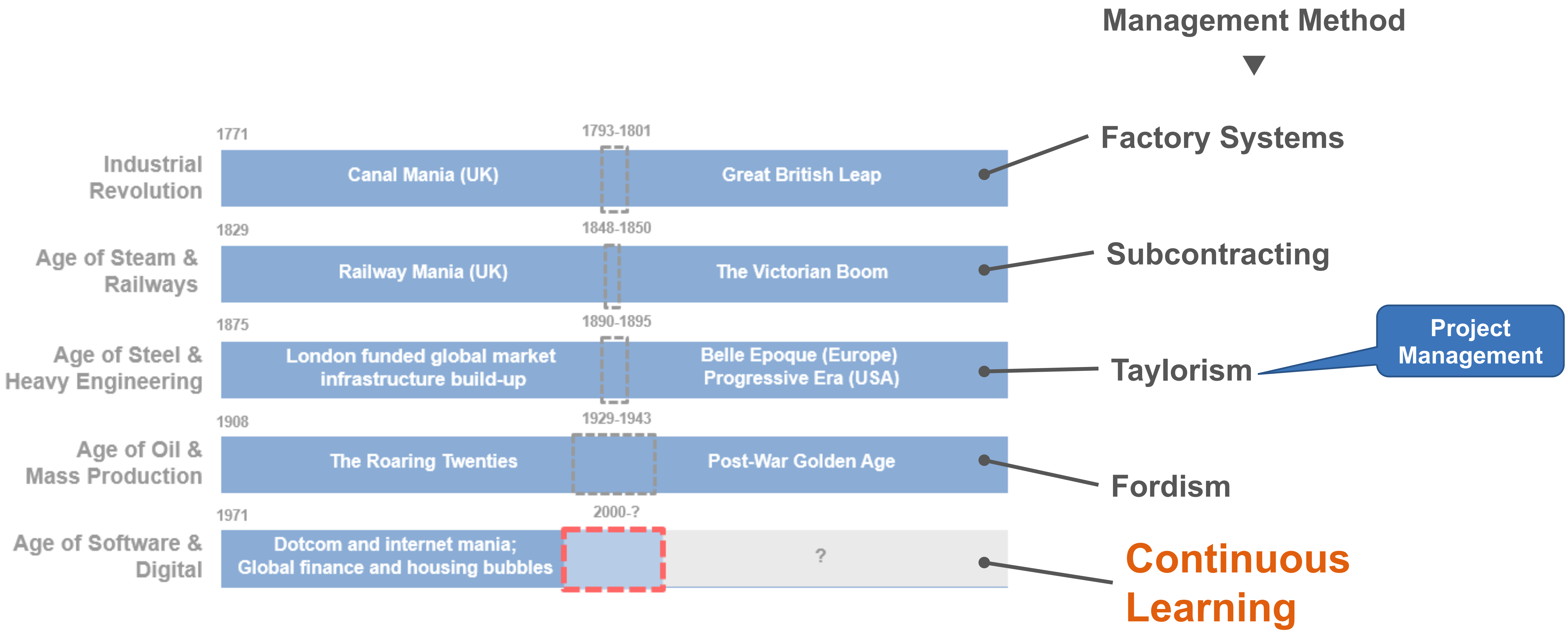
Understand the Value of Continuous Learning

Technological revolutions periodically create a new economic order

		Installation Period	Turning Point	Deployment Period
Industrial Revolution	1771	Canal Mania (UK)	1793-1801	Great British Leap
Age of Steam & Railways	1829	Railway Mania (UK)	1848-1850	The Victorian Boom
Age of Steel & Heavy Engineering	1875	London funded global market infrastructure build-up	1890-1895	Belle Epoque (Europe) Progressive Era (USA)
Age of Oil & Mass Production	1908	The Roaring Twenties	1929-1943	Post-War Golden Age
Age of Software & Digital	1971	Dotcom and internet mania; Global finance and housing bubbles	2000-?	?

Adapted from Technological Revolutions and Financial Capital, Carlota Perez

…and new methods for managing people and work



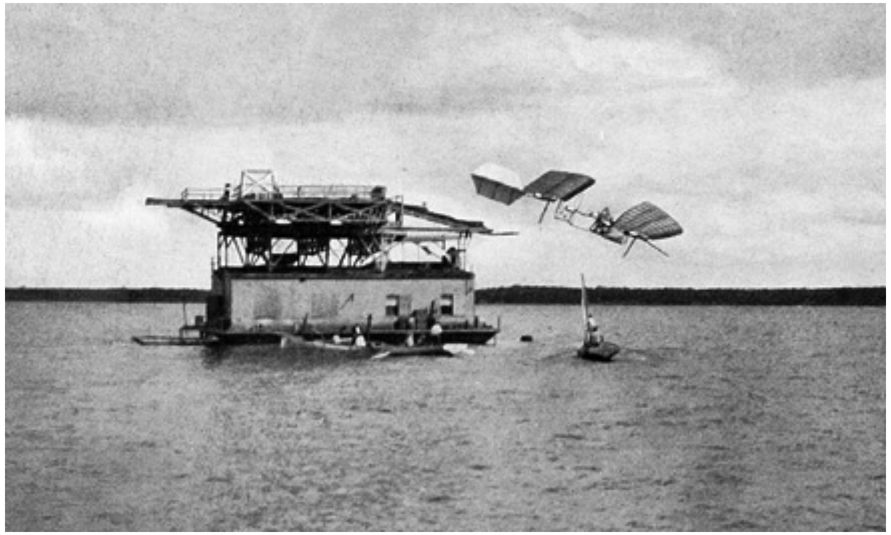
					Management Method
Industrial Revolution	1771	Canal Mania (UK)	1793-1801	Great British Leap	Factory Systems
Age of Steam & Railways	1829	Railway Mania (UK)	1848-1850	The Victorian Boom	Subcontracting
Age of Steel & Heavy Engineering	1875	London funded global market infrastructure build-up	1890-1895	Belle Epoque (Europe) Progressive Era (USA)	Taylorism
Age of Oil & Mass Production	1908	The Roaring Twenties	1929-1943	Post-War Golden Age	Fordism
Age of Software & Digital	1971	Dotcom and internet mania; Global finance and housing bubbles	2000-?	?	Continuous Learning

Project Management

The Race for Flight: A real-life learning challenge

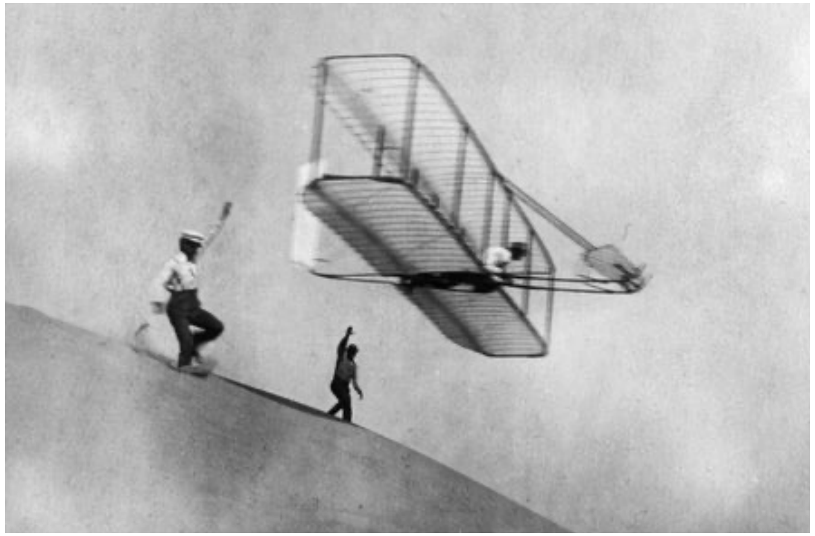
'Big Bang' development

- Samuel Langley
  - Well-funded - $72,000
  - Designed a single-point flying machine
  - Never flew due to fundamental design flaws



Iterative Learning

- Wright Brothers
  - Spent less than $1,000 US
  - Iteratively learned about barriers to flight
  - Created the first flying machine



Explain Agile Product Development

Agile development’s roots are in product development (not software)

- Product development from companies (Fuji-Xerox, Canon, Honda, NEC) shared six characteristics:

1. Built-in instability
2. Self-organizing project [product] teams
3. Overlapping development phases
4. “Multilearning”
5. Subtle control
6. Organizational transfer of learning

HBR

JANUARY–FEBRUARY 1986

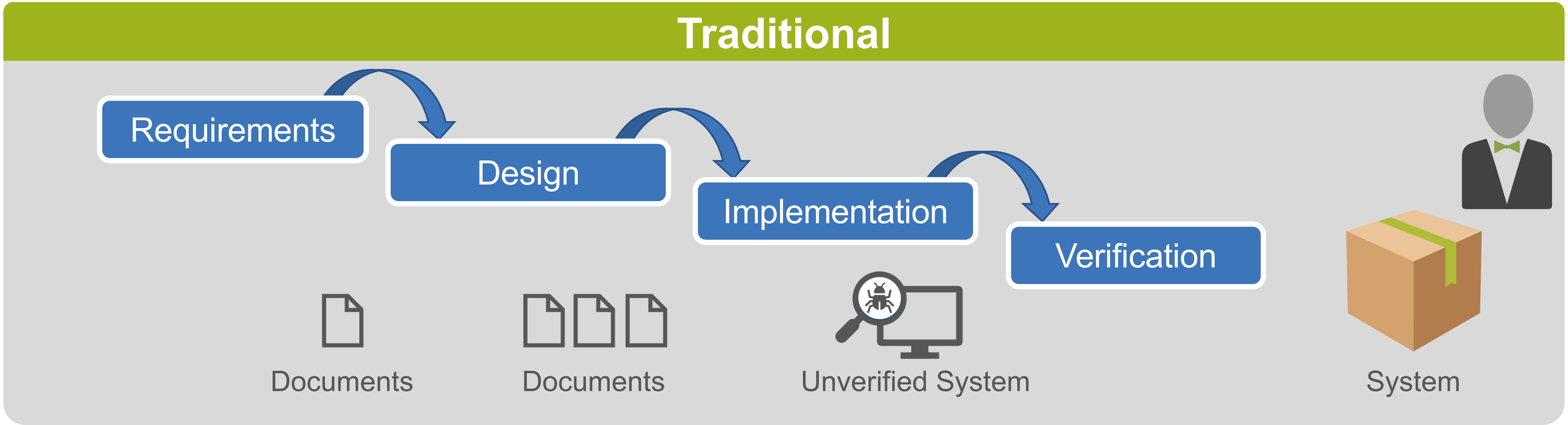
The New New Product Development Game

Hirotaka Takeuchi and Ikujiro Nonaka

[illegible]

—Hirotaka Takeuchi and Ikujiro Nonaka,
“The New New Product Development Game,”
Harvard Business Review, January 1986

Agile Product Development goal: Deliver quick for fast feedback



Traditional

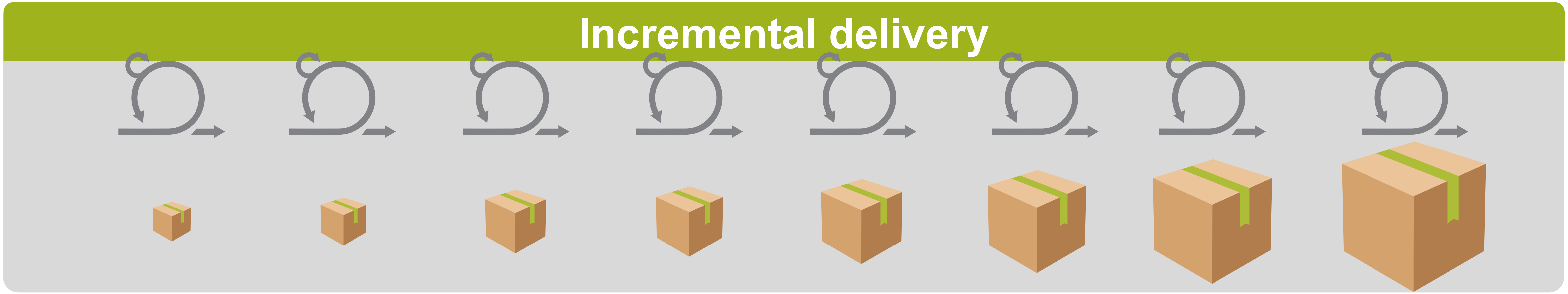
Requirements → Design → Implementation → Verification

Documents

Documents

Unverified System

System

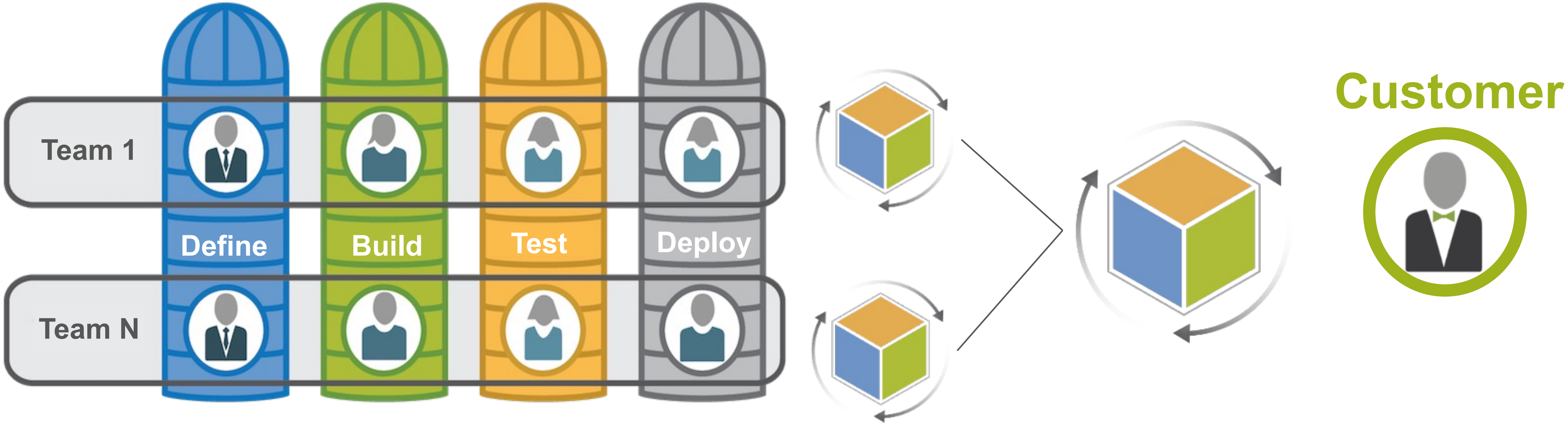


Incremental delivery

Agile Teams are optimized to deliver quickly

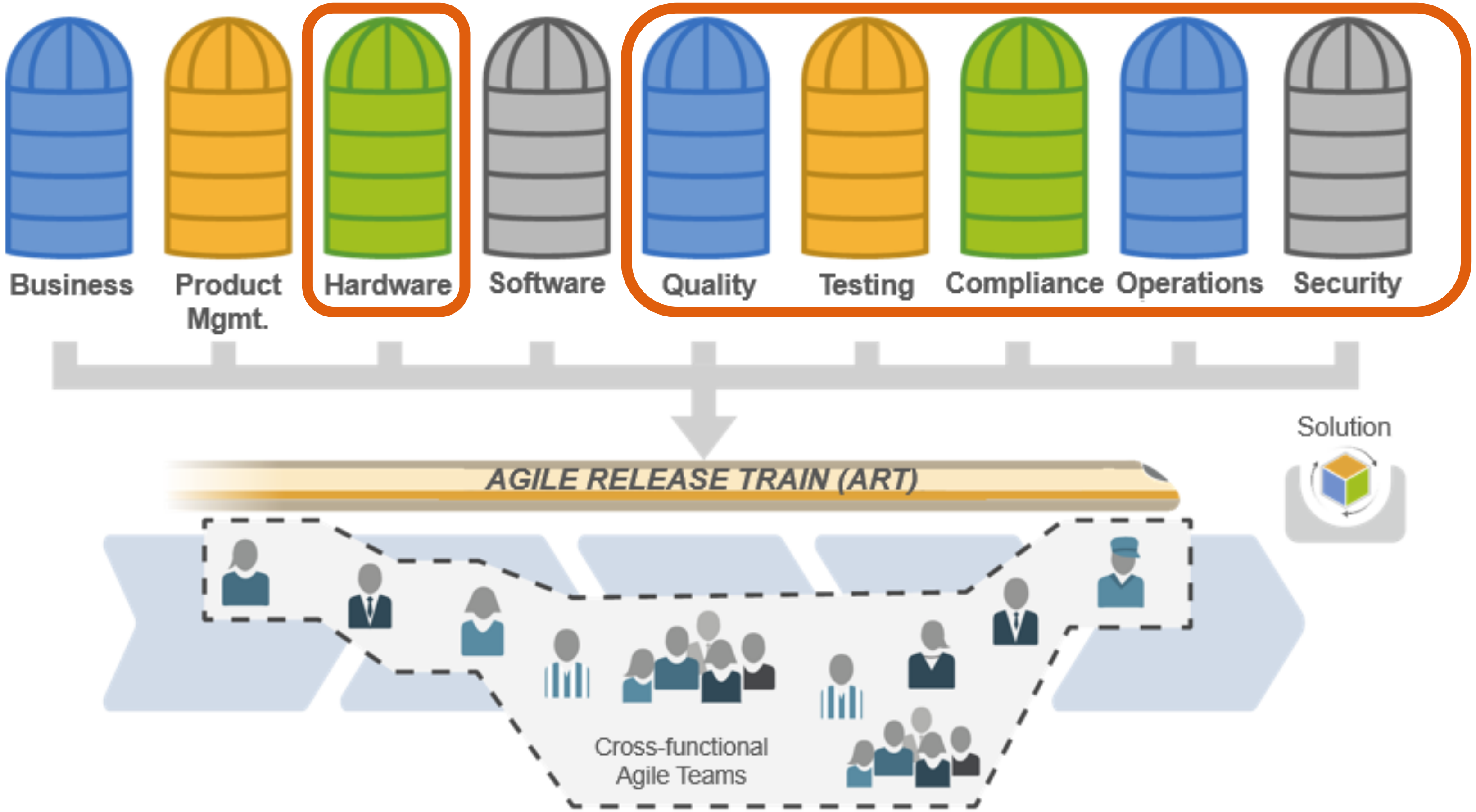
Agile Teams are cross-functional, self-organizing entities that can define, build, test, and where applicable, deploy increments of value.

- All requisite skills and *authority*
- Well-known set of roles, events, artifacts, and practices



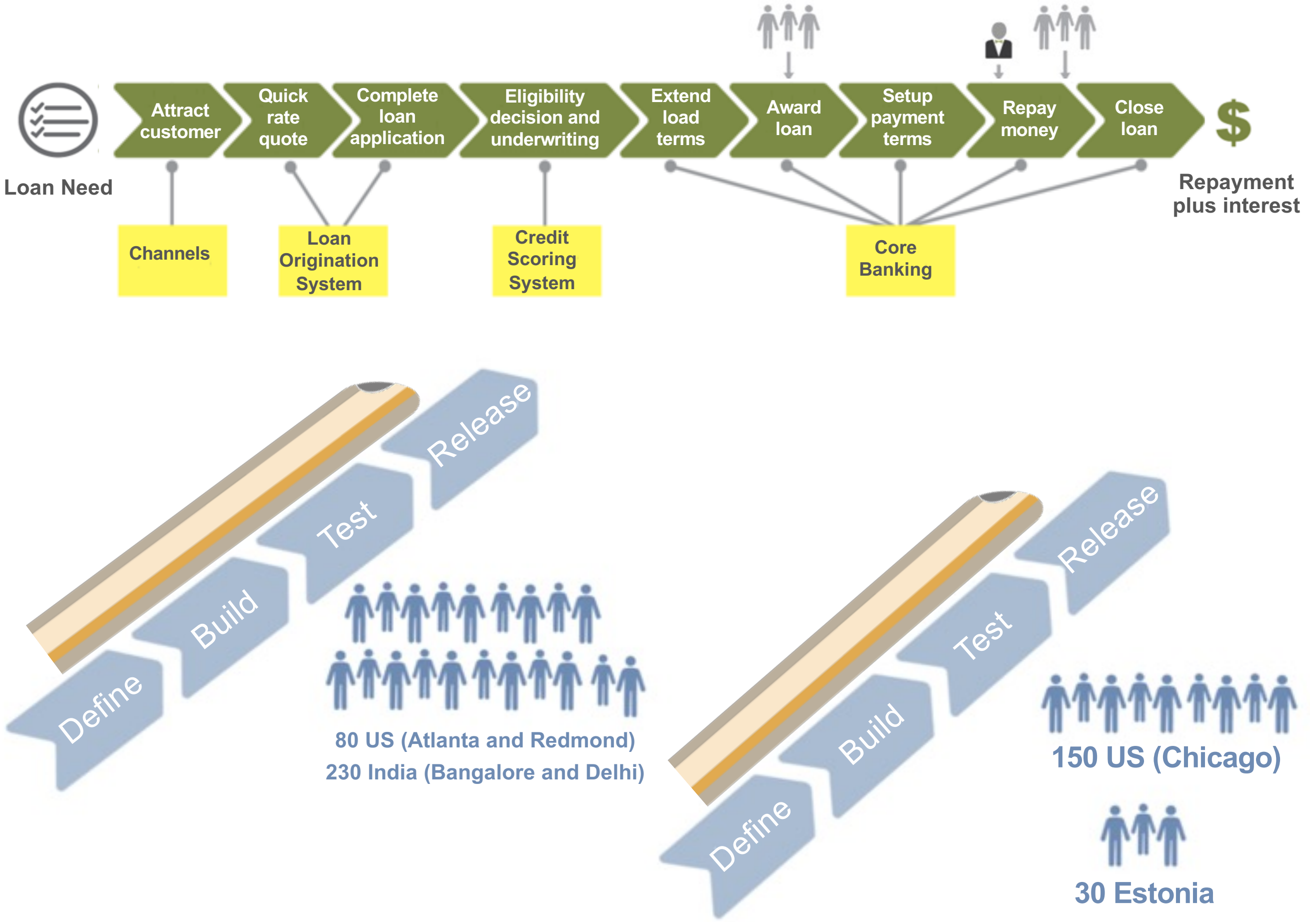
Bigger systems require a team-of-Agile teams

- All requisite *skills and authority* necessary to deliver value
- Scales the well-known set of roles, events, artifacts, and practices



SCALED AGILE®

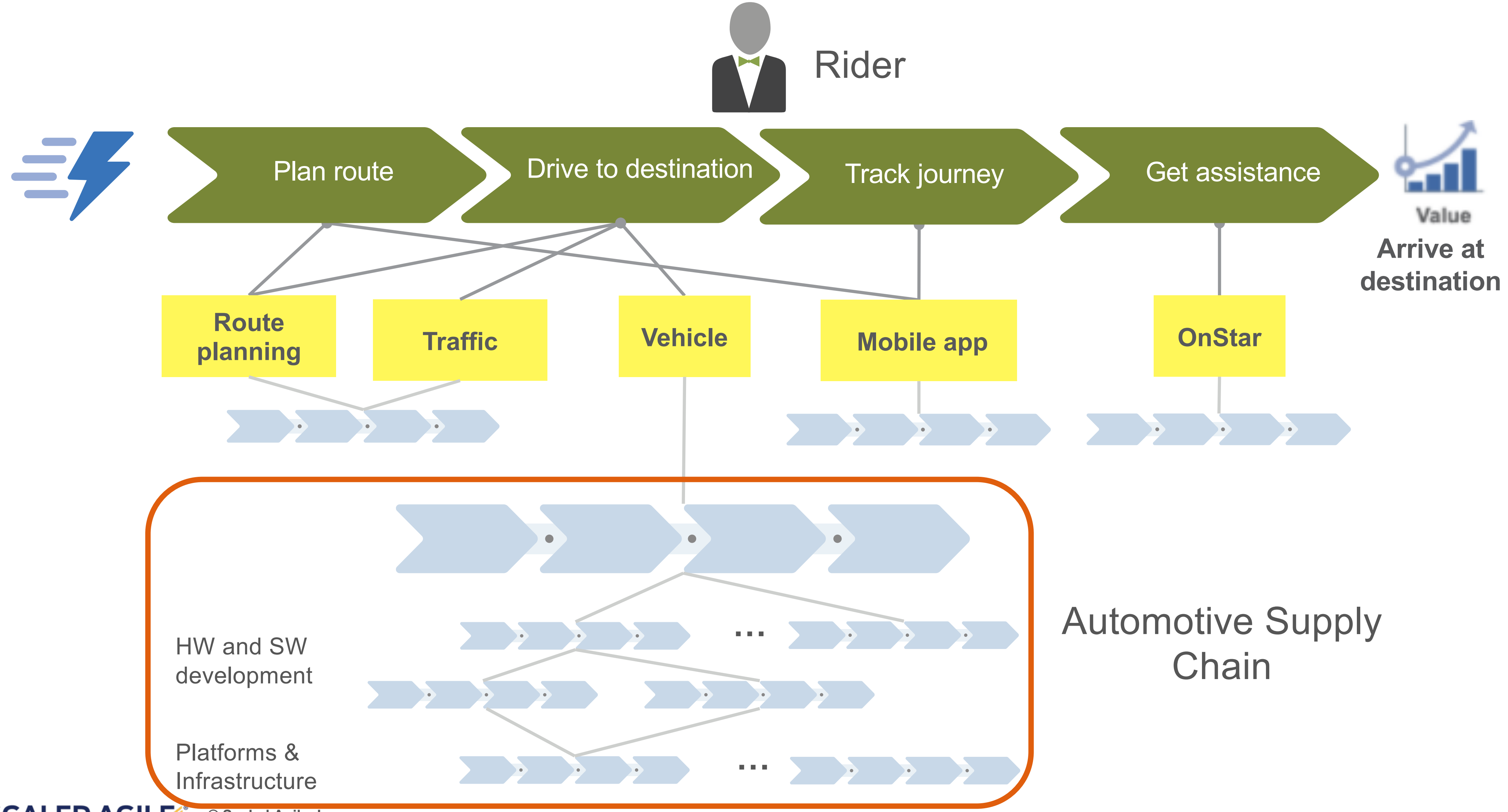
Development value streams (DVS) build products to support operations



1. Operational Value Streams
The sequence of activities needed to deliver a product or service to a Customer. Examples include manufacturing a product, fulfilling an e-commerce order, admitting and treating a patient, providing a loan, and delivering a professional service.

2. Development Value Streams
The sequence of activities needed to convert a business hypothesis into a technology-enabled Solution that delivers Customer value. Examples include designing and developing a medical device, developing and deploying a CRM system, and an eCommerce web site.

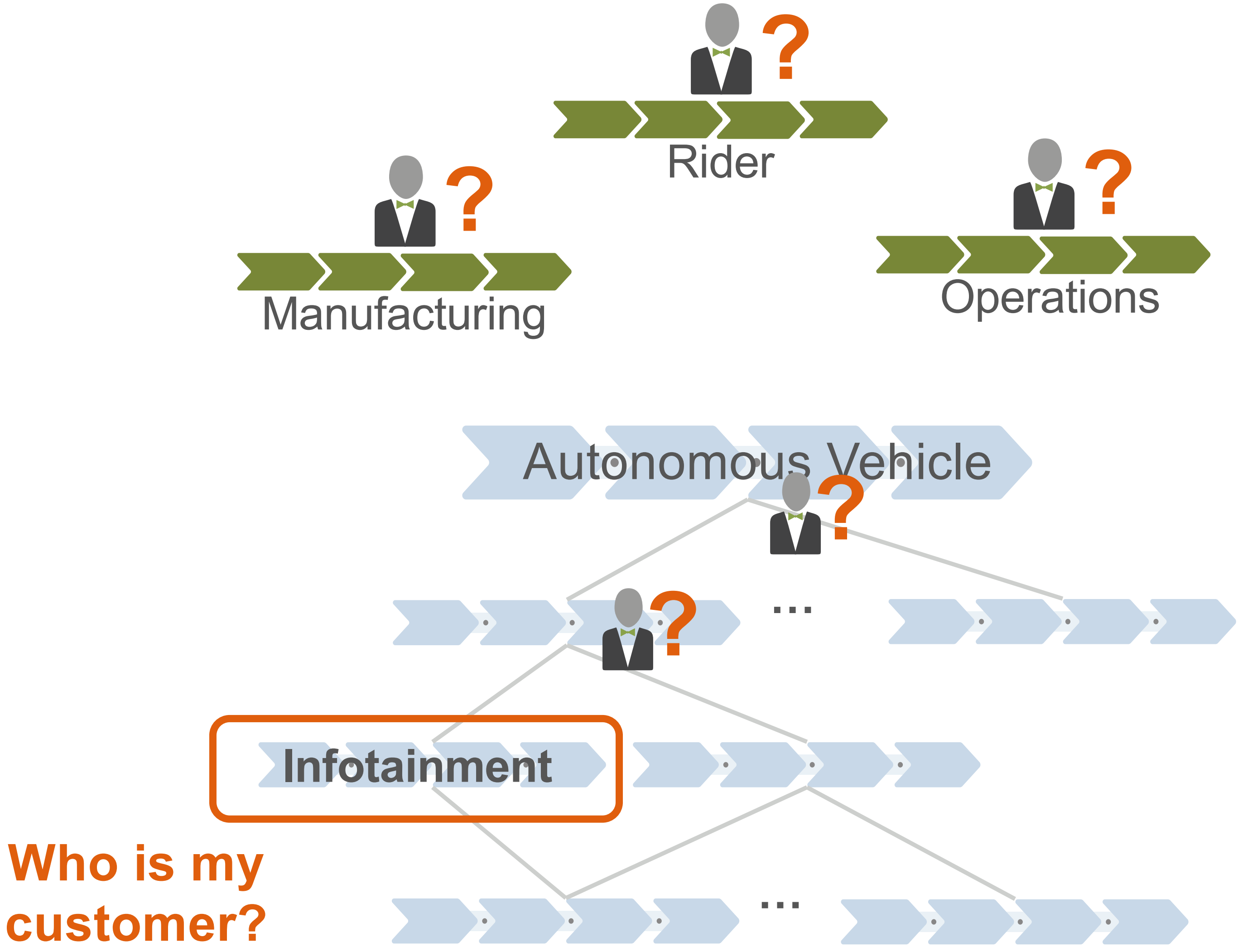
Big systems are built by a network of DVSs



Automotive Supply Chain

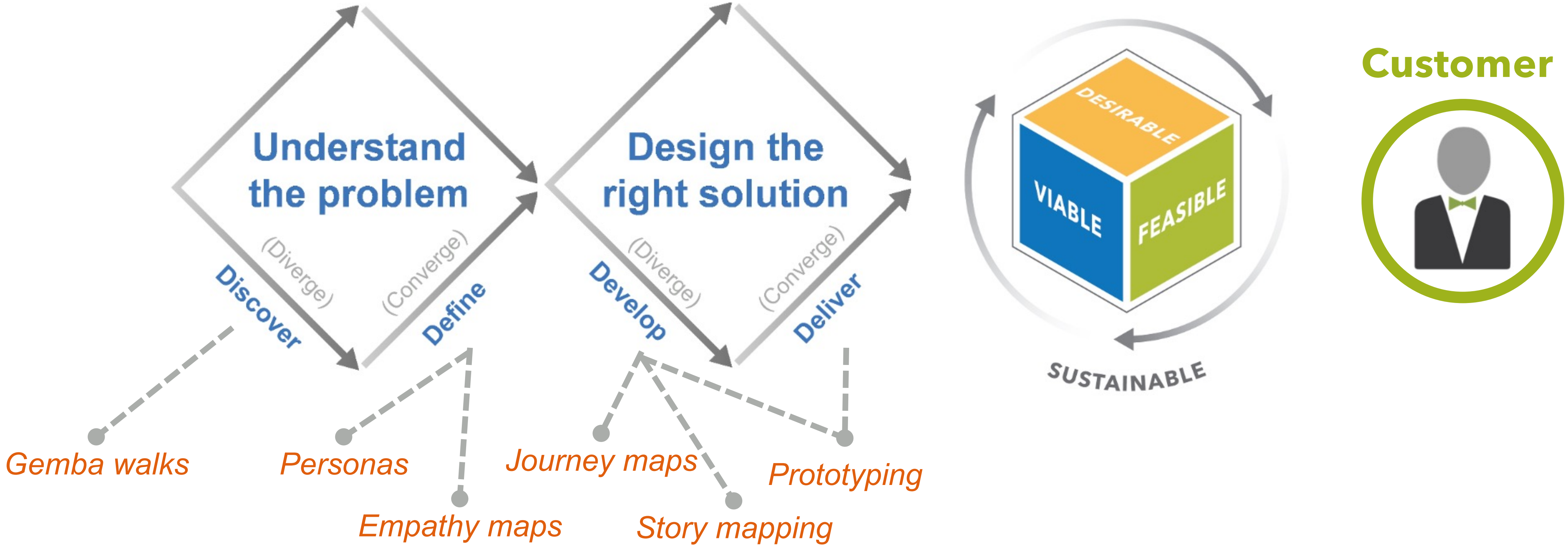
Feedback requires a connection with the Customer

- Supply chains introduce a hierarchy of potential customers
  - Big systems must address multiple OVSs
  - Some DVSs deliver to other DVSs (and OVSs)



Apply Design Thinking to better understand the customer

- Emphasize value delivery over meeting specifications and schedules
- Customers vary – end-users, other developers, operations/maintenance, etc.



Understand the problem

(Diverge) (Converge)

Discover Define

Design the right solution

(Diverge) (Converge)

Develop Deliver

DESIRABLE VIABLE FEASIBLE

SUSTAINABLE

Customer

Gemba walks

Personas

Empathy maps

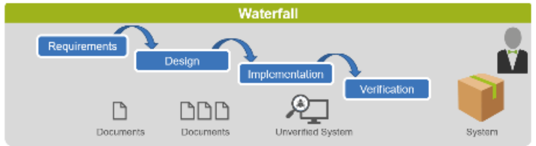
Journey maps

Story mapping

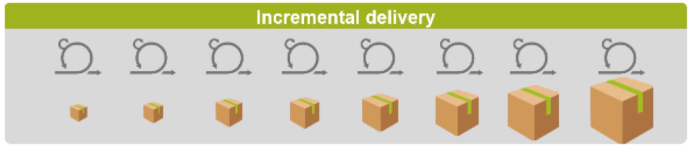
Prototyping

Specify the System Incrementally

Lower the specification batch size to learn faster

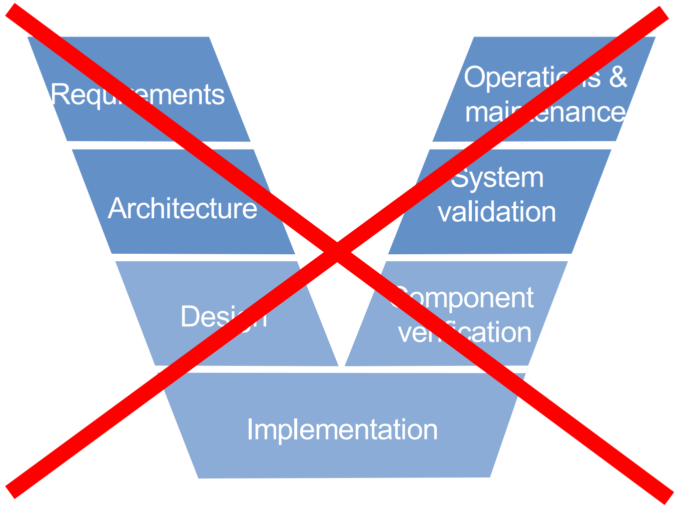


- 100 Shall statements
- Many questions



- 1000 Shall statements
- Many decisions

Traditional ‘V’

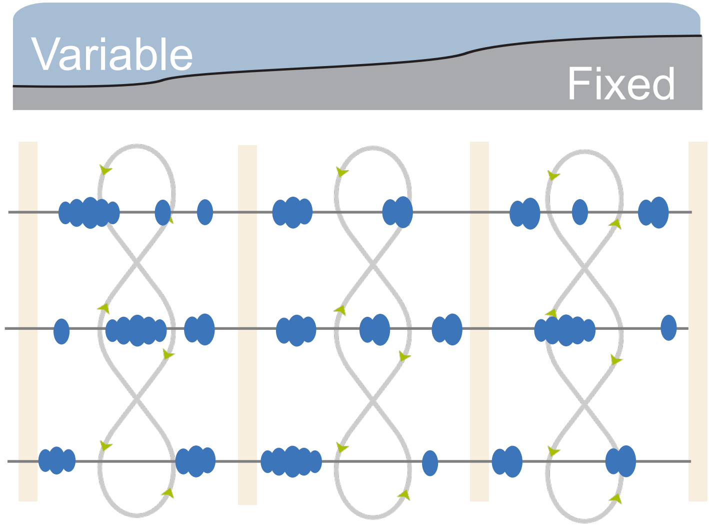


Continuous Flow

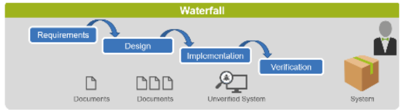
Evolve the Solution Intent

Manage Capabilities and Features

Adjust Roadmaps



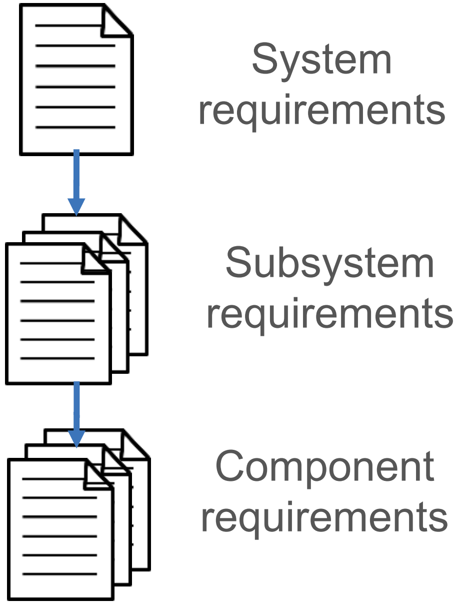
Replace detailed specifications with backlogs and roadmaps



Incremental delivery

Traditional requirements

Provide no opportunity for adjustment based on feedback



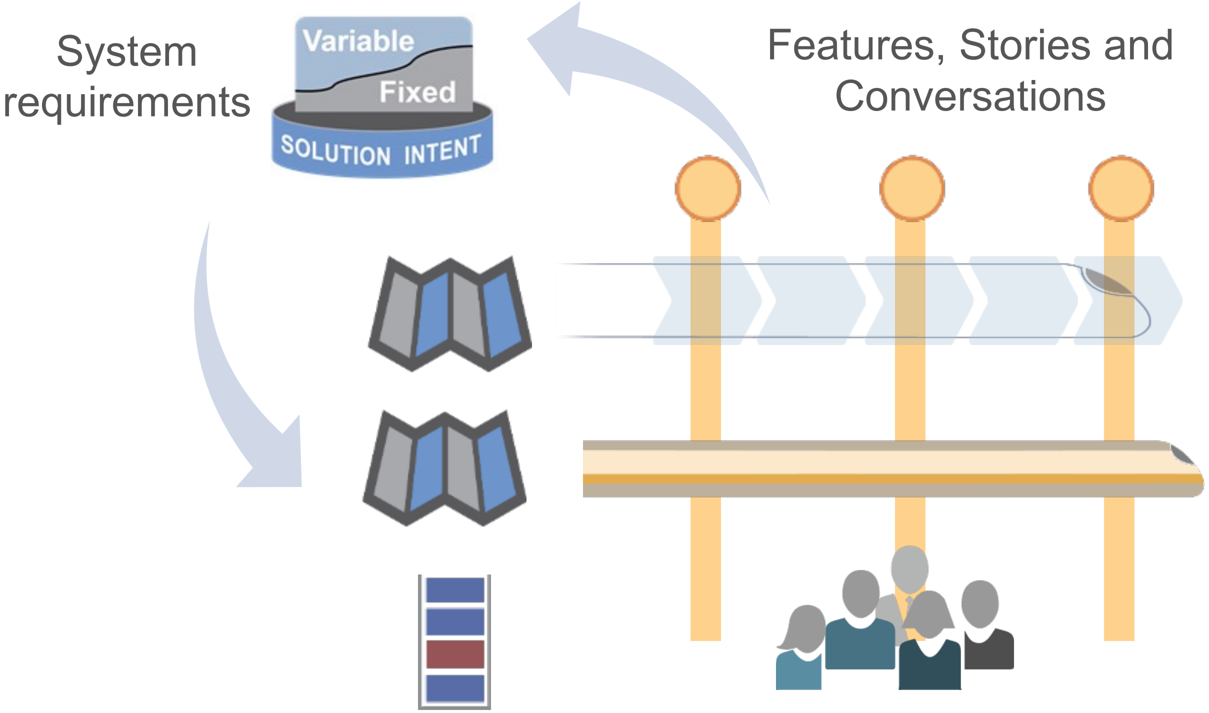
System requirements

Subsystem requirements

Component requirements

Lean-agile requirements

Evolve continually from variable to fixed based on learning



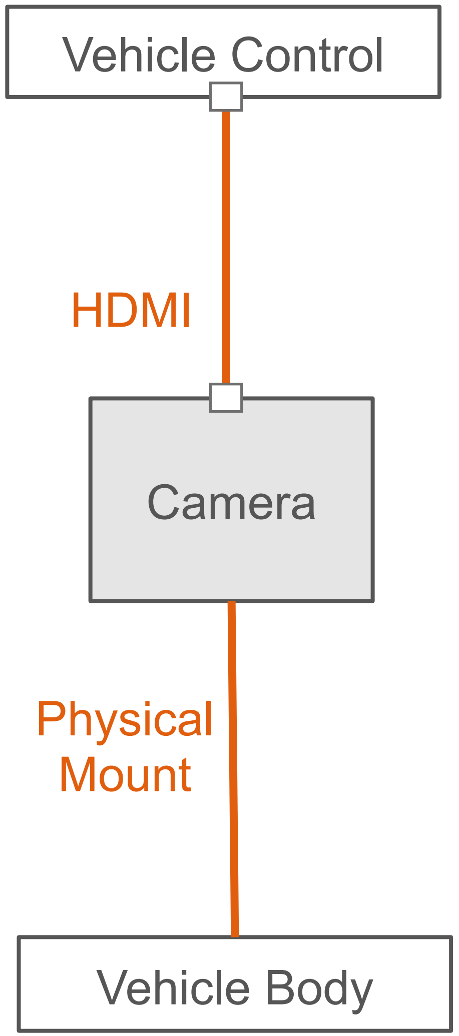
System requirements

Features, Stories and Conversations

Design for Change

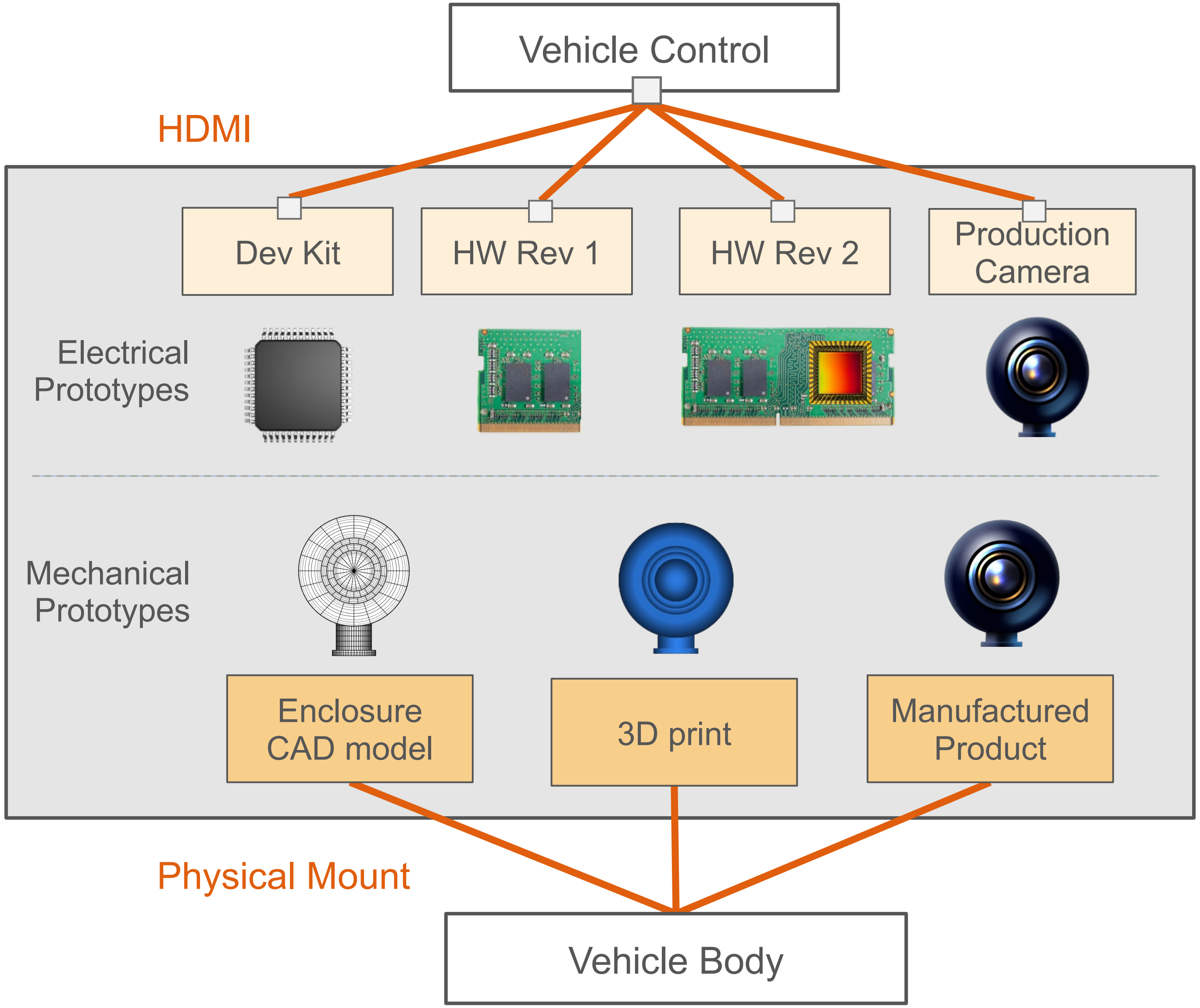
Modular designs with defined interfaces support efficient change

- To evolve designs, define interfaces first
- Interfaces include software APIs, signals, and physical connections
- Interfaces accelerate changes in both development and production environments
- Interfaces enable set-based design



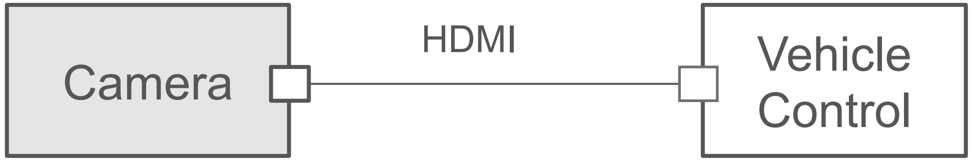
Interfaces enable frequent, independent design iterations

- Allow teams to independently evolve their designs
- Support exploration of independent design sets (SBD)
- Enable frequent integration



Design for change in electrical components

- Favor programmable devices (FPGAs vs. SoCs over ASICs)
- Be intentional about connection permanence (electrical connector vs. solder)

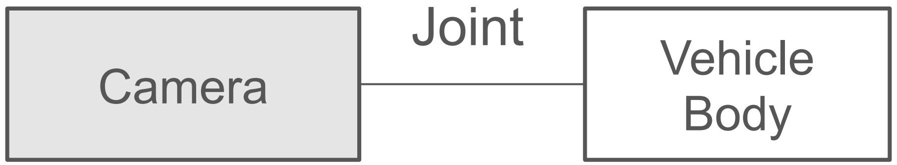


Camera design decisions:

- Should we implement the A/D conversion with an ASIC or a programmable device?
- Should we make the connection a non-permanent plug connector or semi-permanent solder connection?

Design for change in mechanical components

- Component's joints define the mechanical interface
  - May use permanent (weld) or non-permanent (bolt) join
  - Interface may transfer material – force, light, thermal, fluids, etc.
- Consider which linkages should be software-controllable



- How is the joint implemented?
- What material properties are transferred?
- Should the linkage be software-controllable?

Frequently integrate the end-to-end solution

What NASA learned from SpaceX

“SpaceX is good at flying, testing, and even being willing to fail. Then fly, test, fail, fix. And they can reiterate this over and over extremely fast. *The willingness to fail is something NASA has frankly lacked, but it’s what enables SpaceX to move so fast. To rapidly iterate and improve.*

NASA has a history of qualifying every component and subcomponent. Every piece of every rocket is full qualified, and everything must go perfect on every launch. And that slows us down.”

-- Jim Bridenstine, NASA Chief Administrator

KENNEDY SPACE CENTER

Exclusive interview with Elon Musk and Jim Bridenstine about #DM2, SpaceX's first crewed launch!

Everyday Astronaut

948K subscribers

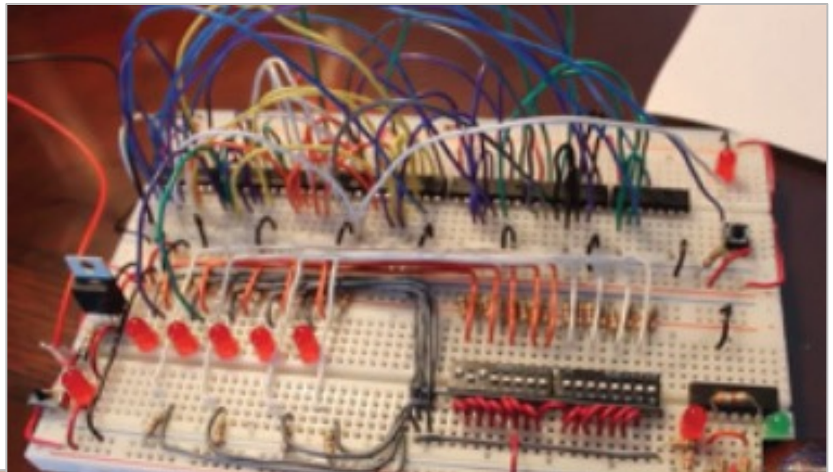
https://youtu.be/p4ZLysa9Qqg?t=266

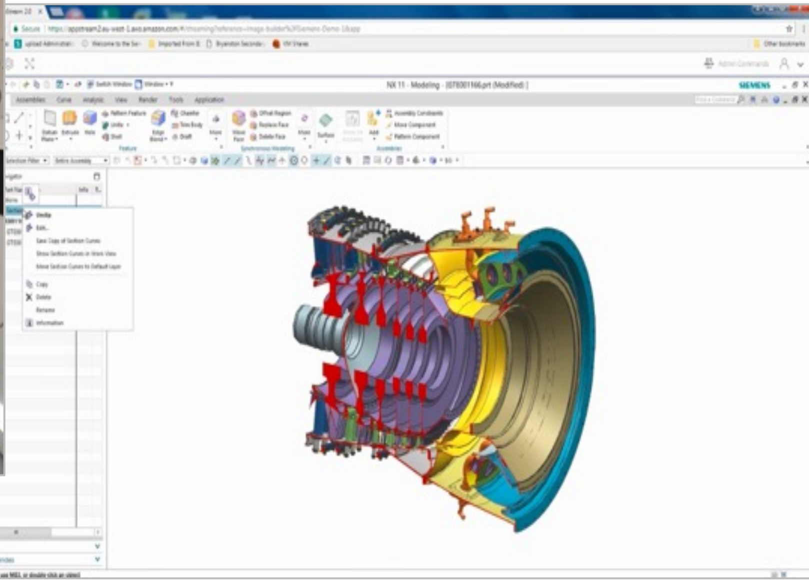
What would it take to build and operationally test a solution increment every 2-3 months? What value would it provide?

Hardware domains experiment during development

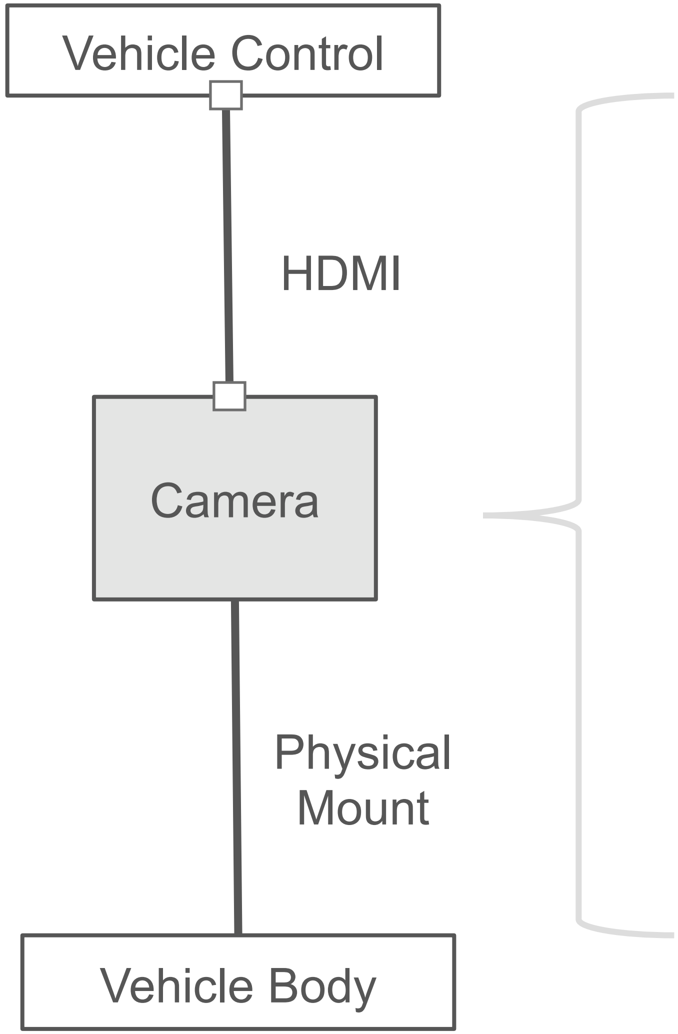
- Provides knowledge and feedback earlier in product lifecycle
- Mitigates risk by validating assumptions sooner

$$\hat{H} = \sum_{n=1}^{N} \frac{\hat{p}_n^2}{2m_n} + V(x_1, x_2, \cdots x_N)$$
$$= -\frac{\hbar^2}{2} \sum_{n=1}^{N} \frac{1}{m_n} \frac{\partial^2}{\partial x_n^2} + V(x_1, x_2, \cdots x_N)$$

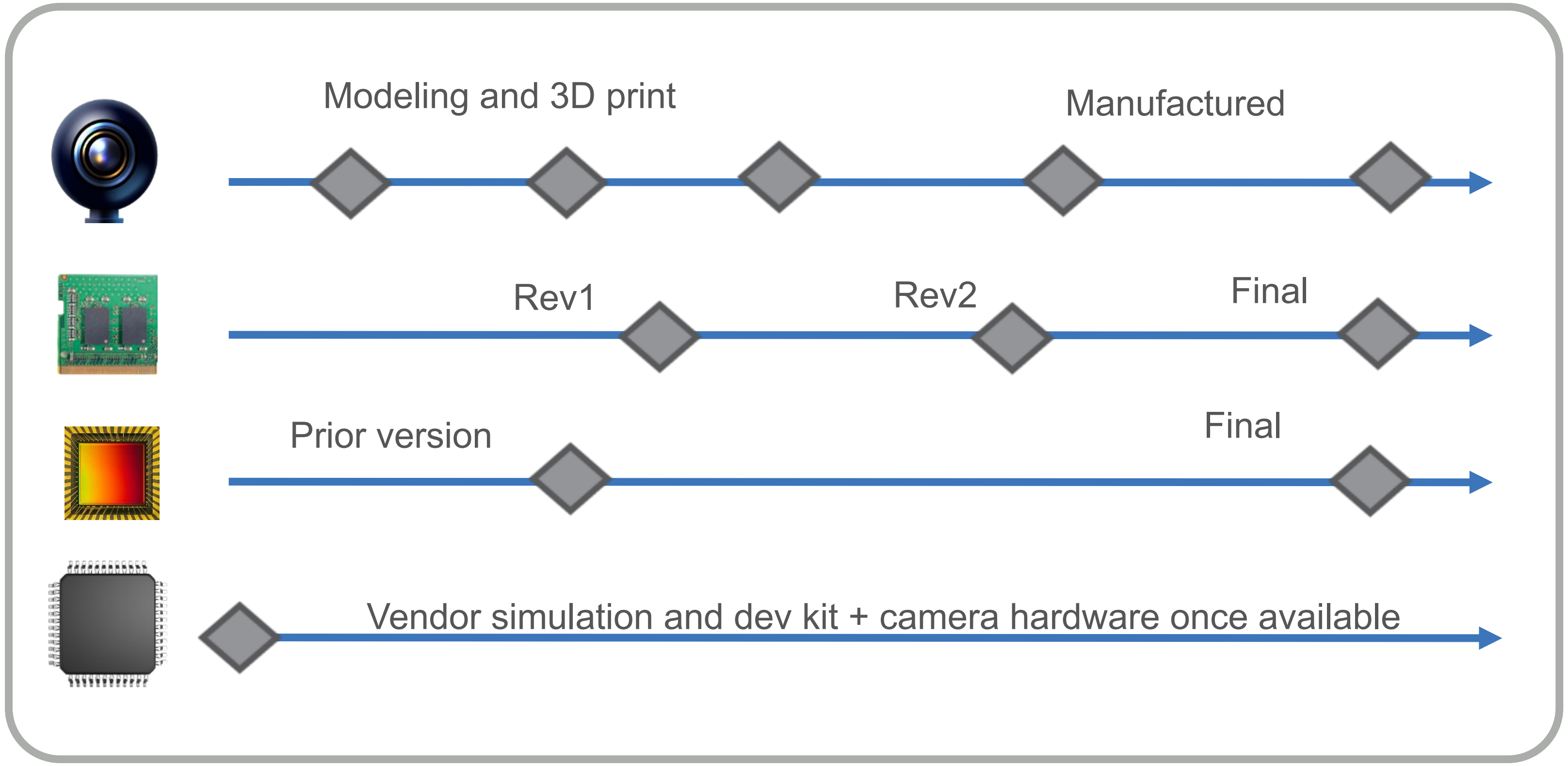




Learning is often done in a local context



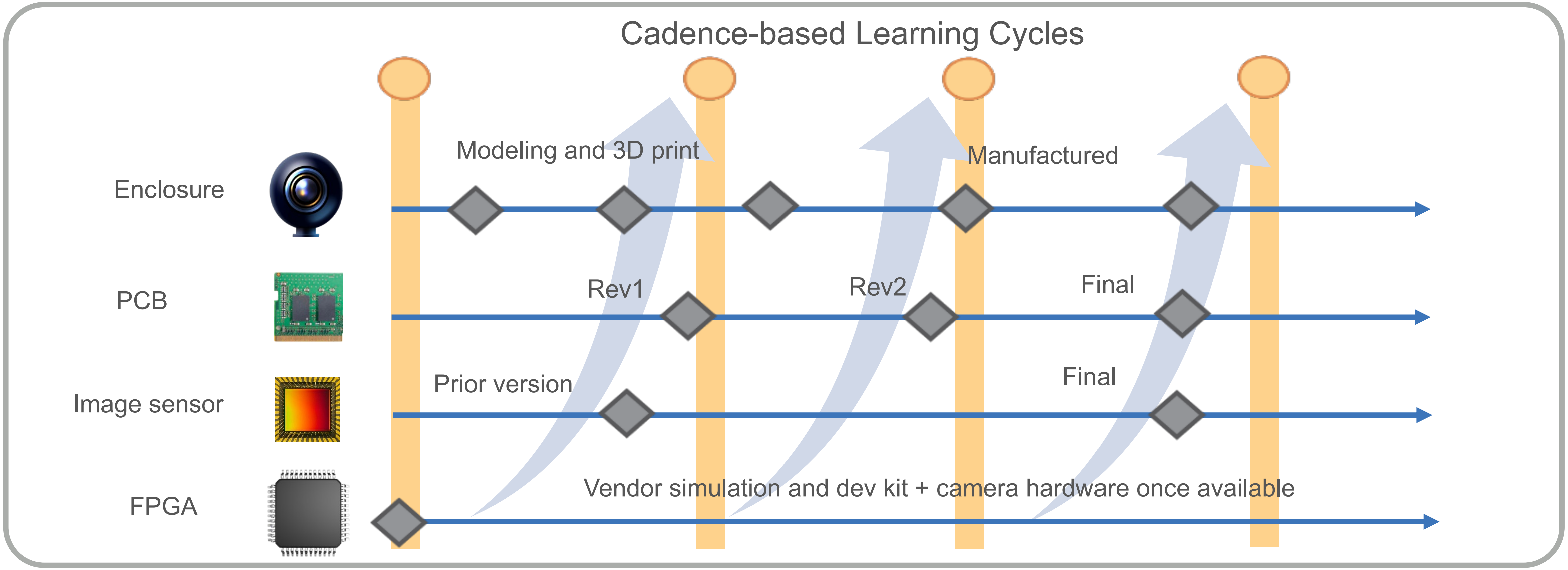
Domain-specific innovations



A common cadence ensures that the entire system is learning

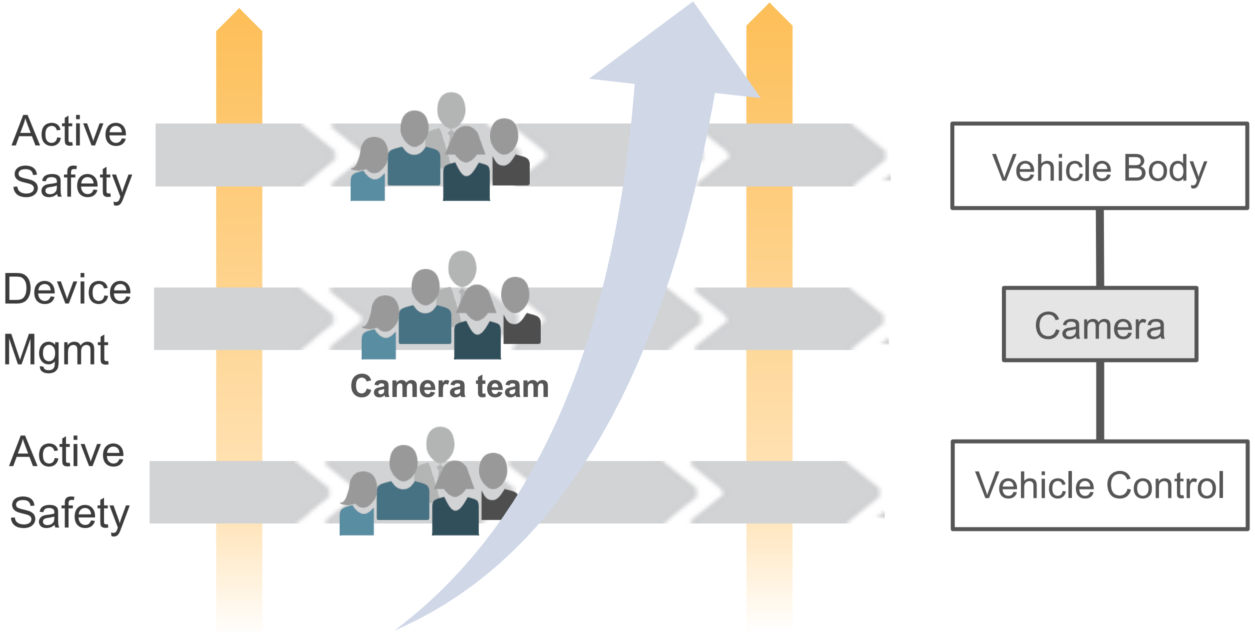
“Integration points control product development and are the leverage points to improve the system. When timing of integration points slips, the project is in trouble.”

- Dantar P. Oosterwal



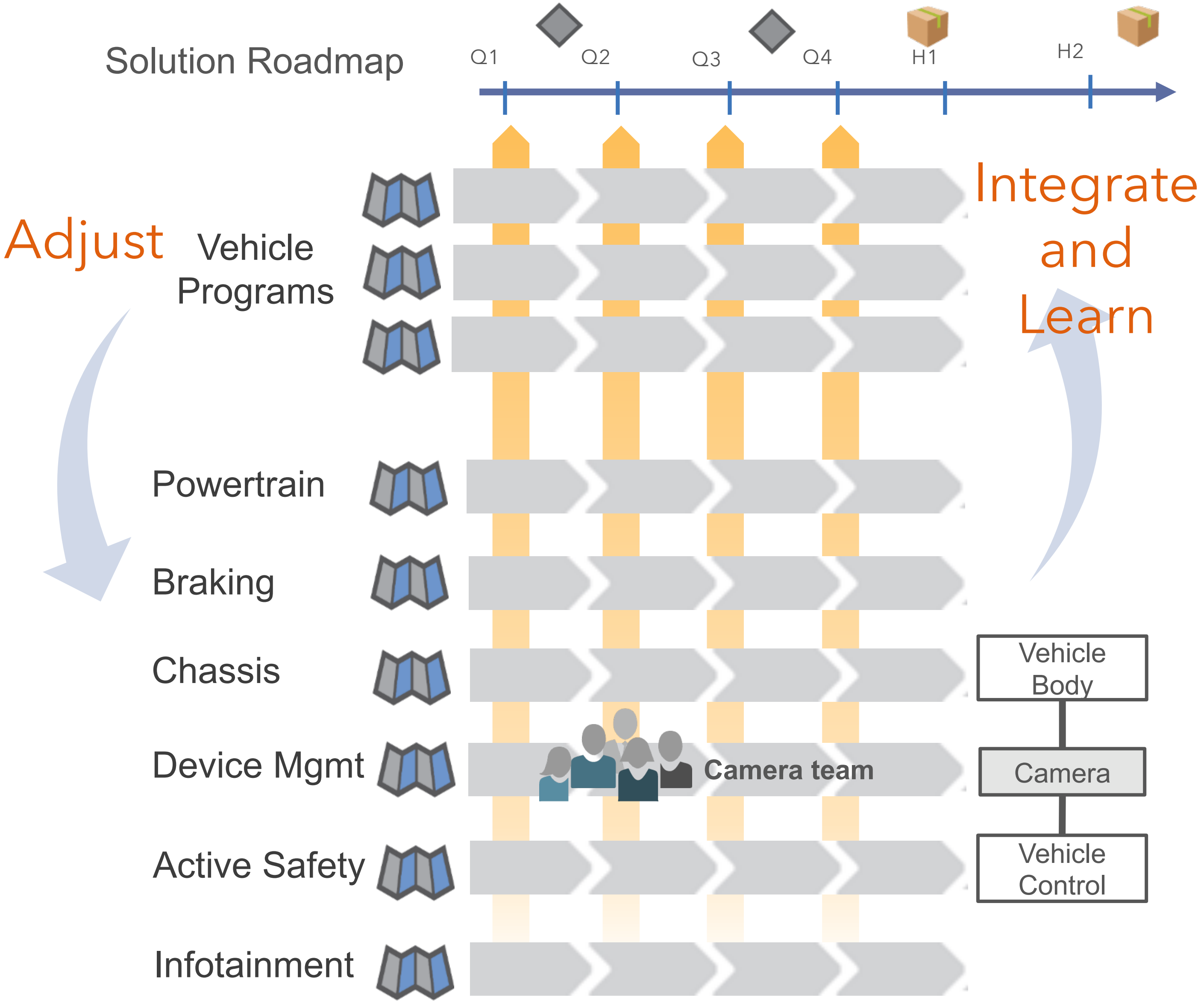
Continu-ishly integrate the end-to-end system

- Frequently integrate changes into richer contexts for frequent verification and validation



Align cadence and roadmaps across the entire system development

- Ensure the system is learning, not just individual teams

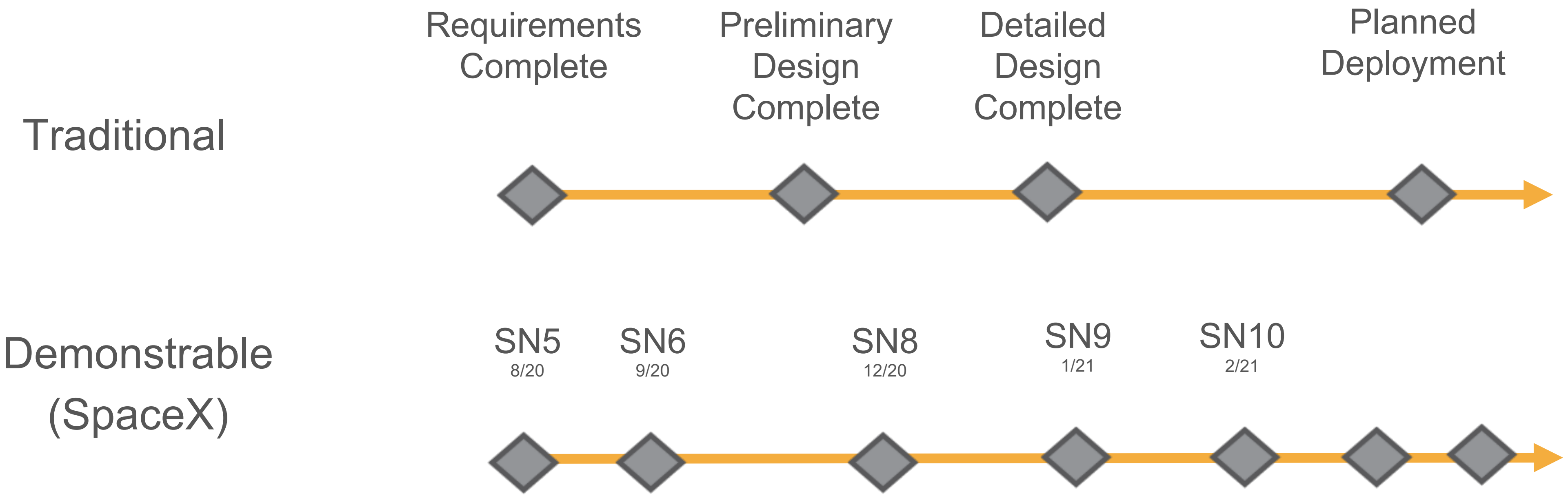


Shift Learning Left

Base milestones on objective evaluation of working systems

Development is more dependent on what needs to be learned than on what tasks must be completed to exit a gate. – Allan Ward

- Fixed, upfront specifications and schedules don't work in environments with high uncertainty and variability



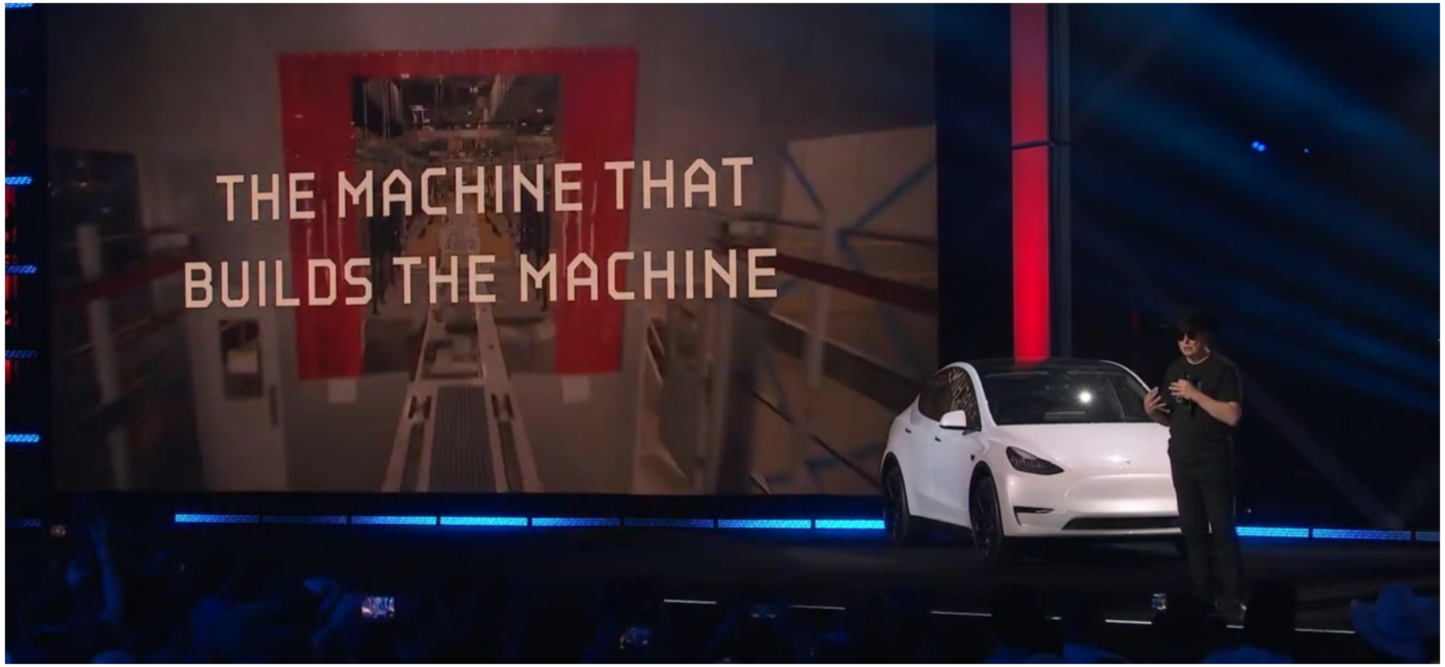
Emphasizing learning create business value

	Falcon 9 Block 1	Falcon 9 Block 2	Falcon 9 Block 3	Falcon 9 Block 4	Falcon 9 Block 5
Year	2010-13	2013-15	2015-17	2017-18	2018-20
Engine	Merlin 1C	Merlin 1D	Merlin 1D	Merlin 1D	Merlin 1D
Innovation	Tried Parachute recovery (failed)	60% More Thrust	17% more thrust First reusable 1st stage	Improved 2nd Stage Engine Thrust upgrades	Solve reuse & reliability
SpaceX NASA Launches	5	15	25	11	27
All Other NASA Launches	23	18	14	11	2

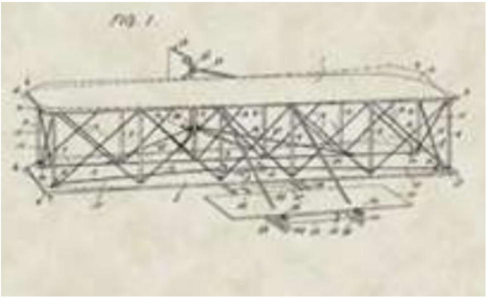
Build the solution and the continuous delivery pipeline (CDP) together

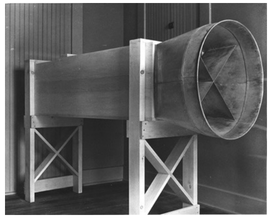
This is the machine that builds the machine and it’s the latest version of the machine that builds the machine. The factory is the product. -- Elon Musk

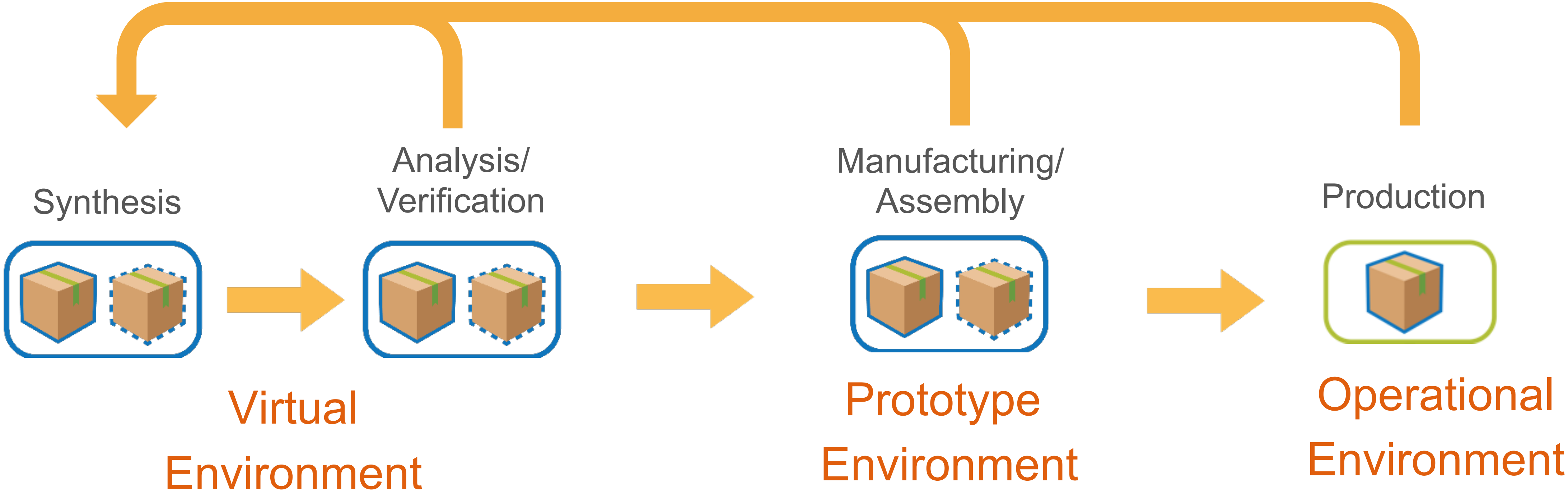
- Products are never one-and-done
- Change the mindset to build quickly and evolve instead of build once and maintain



For hardware, learning occurs in three environments







Synthesis

Analysis/
Verification

Manufacturing/
Assembly

Production

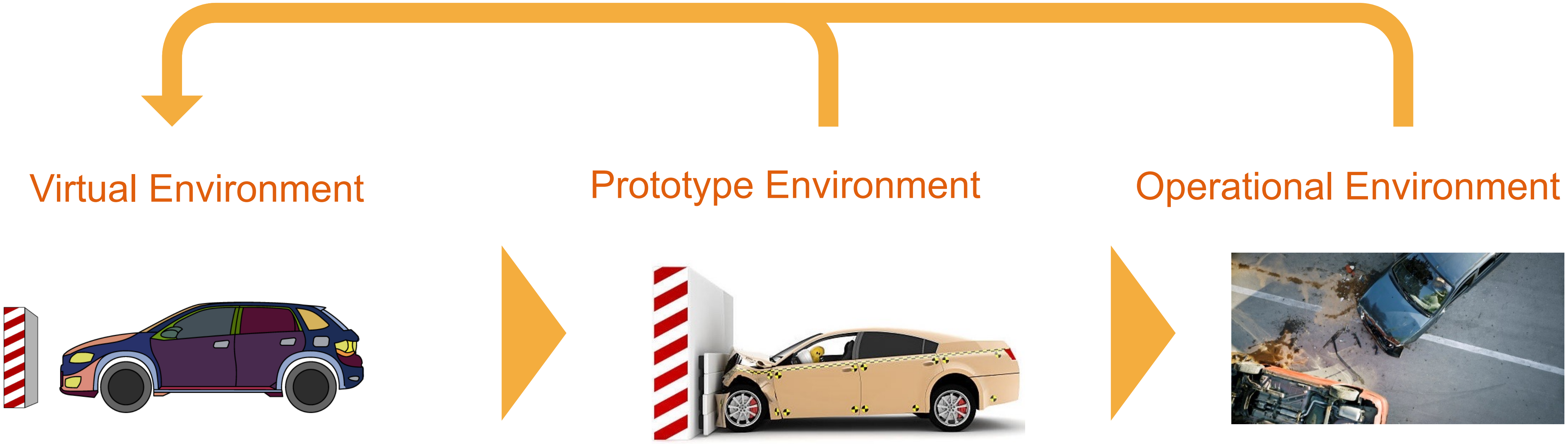
Virtual
Environment

Prototype
Environment

Operational
Environment

The physical and operational worlds validate the virtual one

- Use physical milestones to validate the virtual assumptions



Virtual Environment

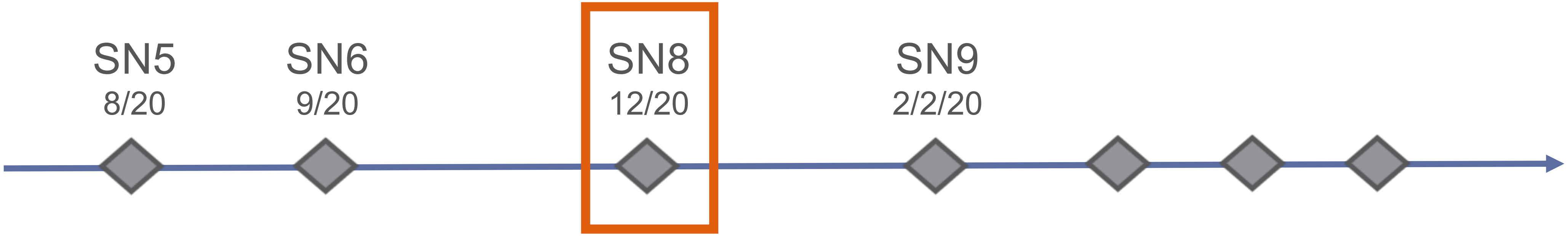
Prototype Environment

Operational Environment

“The simulations are so accurate the [US] government now accepts them”

—Joe Justice, Wikispeed

Provide the psychological safety to learn

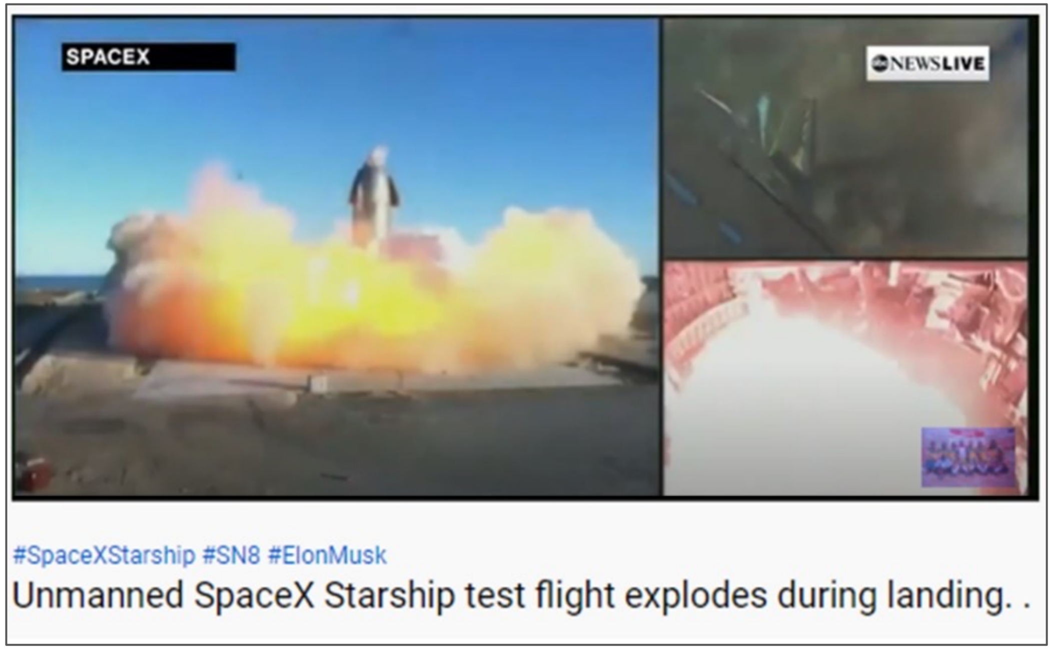


SN5
8/20

SN6
9/20

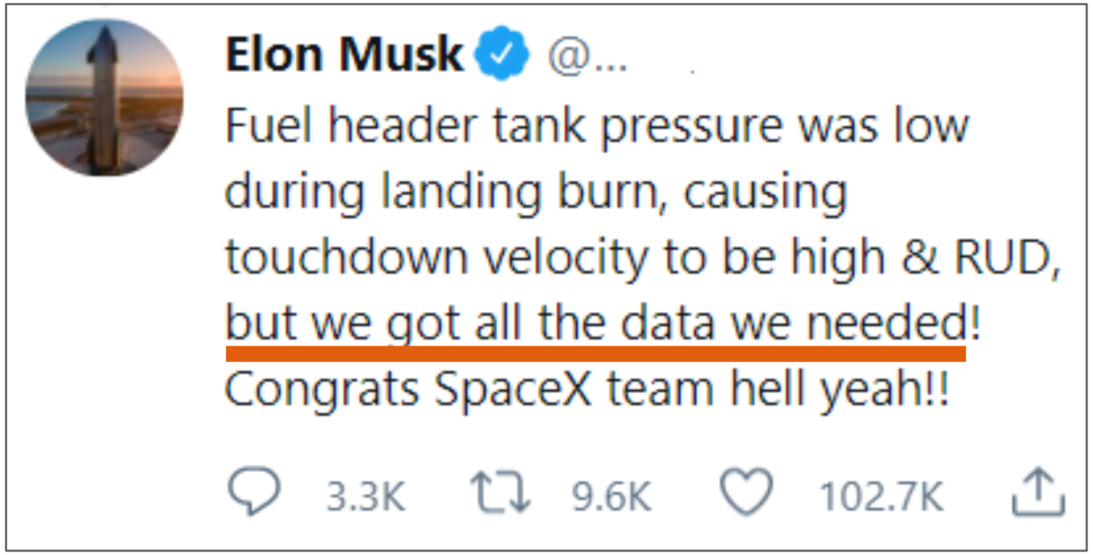
SN8
12/20

SN9
2/2/20



#SpaceXStarship #SN8 #ElonMusk

Unmanned SpaceX Starship test flight explodes during landing. .



Elon Musk @...

Fuel header tank pressure was low during landing burn, causing touchdown velocity to be high & RUD, but we got all the data we needed! Congrats SpaceX team hell yeah!!

3.3K 9.6K 102.7K

Questions