DCDP SAFe For Teams Training

Clear your tables

Introductions

Your role

Your experience with agile

What do you do today?

What part do you play in IOP?

5 = Expert

4 = Very experienced

3 = Experienced

2 = Some experience

1 = Very little/no experience

Daniel Spencer – Enterprise Agile Coach with Eliassen Group

What I Do

- Help organizations succeed with agile at scale
- Develop high-performing teams

My Experience

- 25+ years experience building enterprise software
- 12 years of agile practice
- A bit of STB development - but don't hold me to it!

Prior Roles

- Director of Software Development
- Product Manager
- Scrum Master
- Software Developer
- Project Manager

Agile Experience

- Program Coach
  - DirecTV – DFW, DCDP
  - Thales
- Trainer - Honeywell
- Assessments – VFC, Patelco
- SAFe SPC 4.0, CSM

Why am I here?

01 Provide Foundation Agile, Scrum and SAFe Education

02 Explain, Educate, Answer, Advise

03 Shake up your minds a bit so new information can get in there

04 Challenge you to look at and think about things differently

05 Have FUN!!

SAFe 4.5

SAFe® for Teams

Empowering teams to deliver value in the SAFe enterprise

V4.5.0.1

SAFe® Course Attending this course gives students access to the SAFe Practitioner exam and related preparation materials.

SCALED AGILE®

SAFe® for Teams

1. Introducing the Scaled Agile Framework
2. Building an Agile Team
3. Planning the Iteration
4. Planning the Iteration
5. Executing the PI

Logistics

- Class times
- Breaks
- Lunch
- Restrooms
- Other

To perform the role of a SAFe Practitioner, you should be able to:

- Apply SAFe to scale Lean and Agile development in your enterprise
- Know your team and its role on the Agile Release Train
- Know all other teams on the train, their roles, and the dependencies between the teams
- Plan iterations
- Execute iterations and demonstrate value
- Plan program increments
- Integrate and work with other teams on the train

Lesson 1
Introducing the Scaled Agile Framework

1. Introducing the Scaled Agile Framework
2. Building an Agile Team
3. Planning the Iteration
4. Executing the Iteration
5. Executing the PI

SAFe® Course Attending this course gives students access to the SAFe Practitioner exam and related preparation materials.

Learning objectives

1.1 Connect with the Scaled Agile Framework

1.2 Explore Lean, the Agile Manifesto, and SAFe Principles

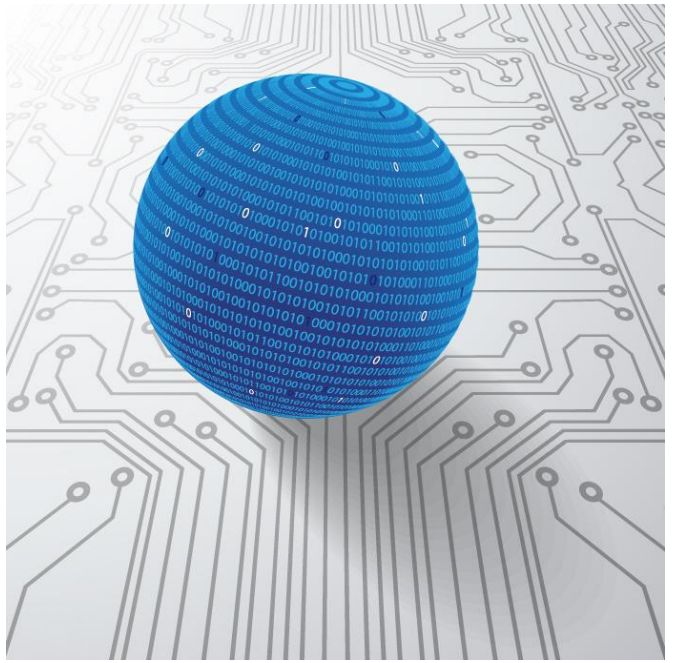
1.3 Identify Scrum, Kanban and XP Practices

1.1 Connect with the Scaled Agile Framework

How do we keep pace?

Our development methods must keep pace with an increasingly complex world.

- We've had Moore's Law for hardware, and now software is eating the world
- Our development practices haven't kept pace; Agile shows the greatest promise, but was developed for small teams
- We need a new approach, one that harnesses the power of Agile and Lean, and applies to the needs of the enterprises who build the world's most important software and systems.

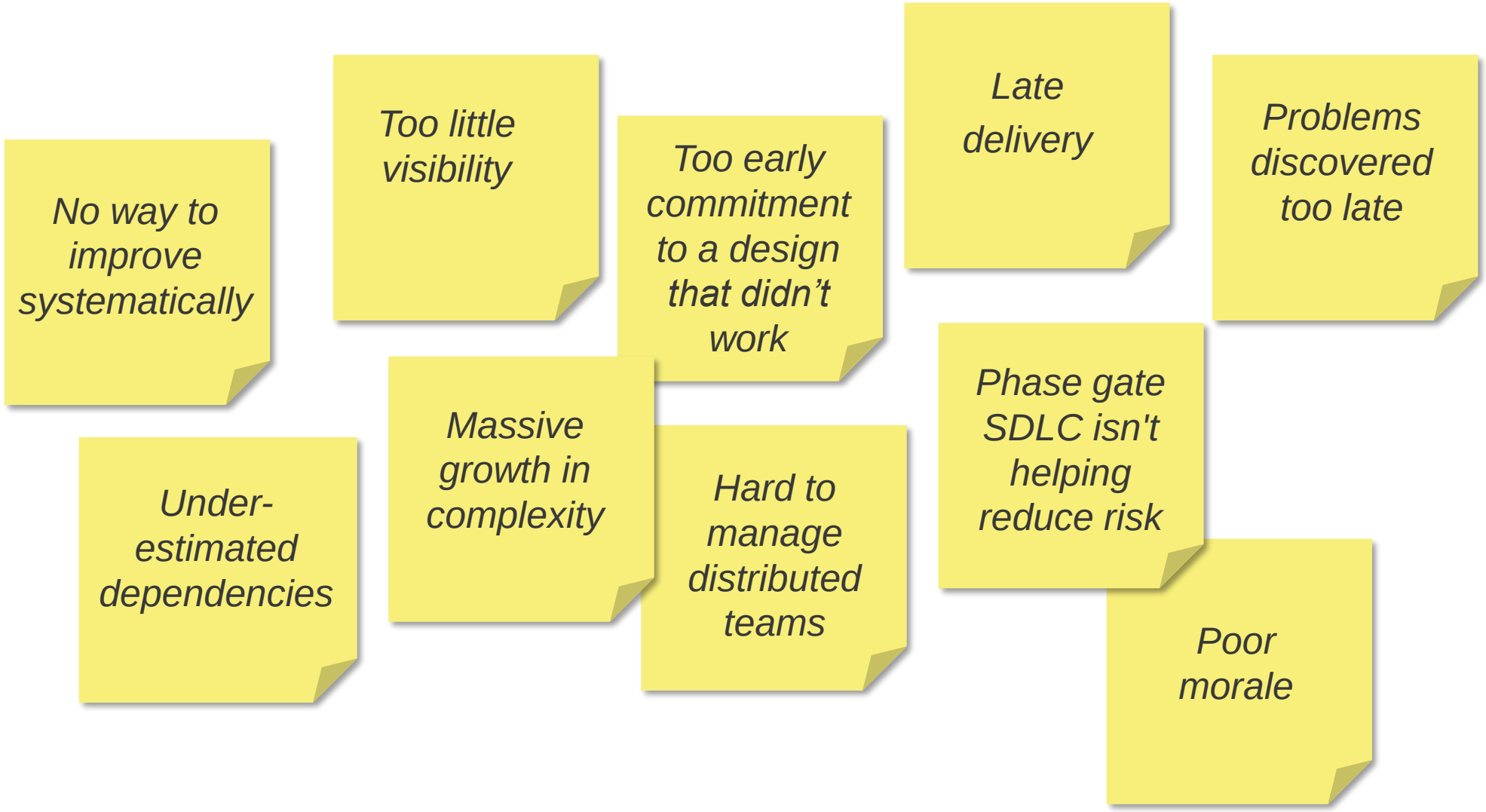


We thought we'd be developing like this.

But sometimes it feels like this.

Library of Congress

And our retrospectives read like this:



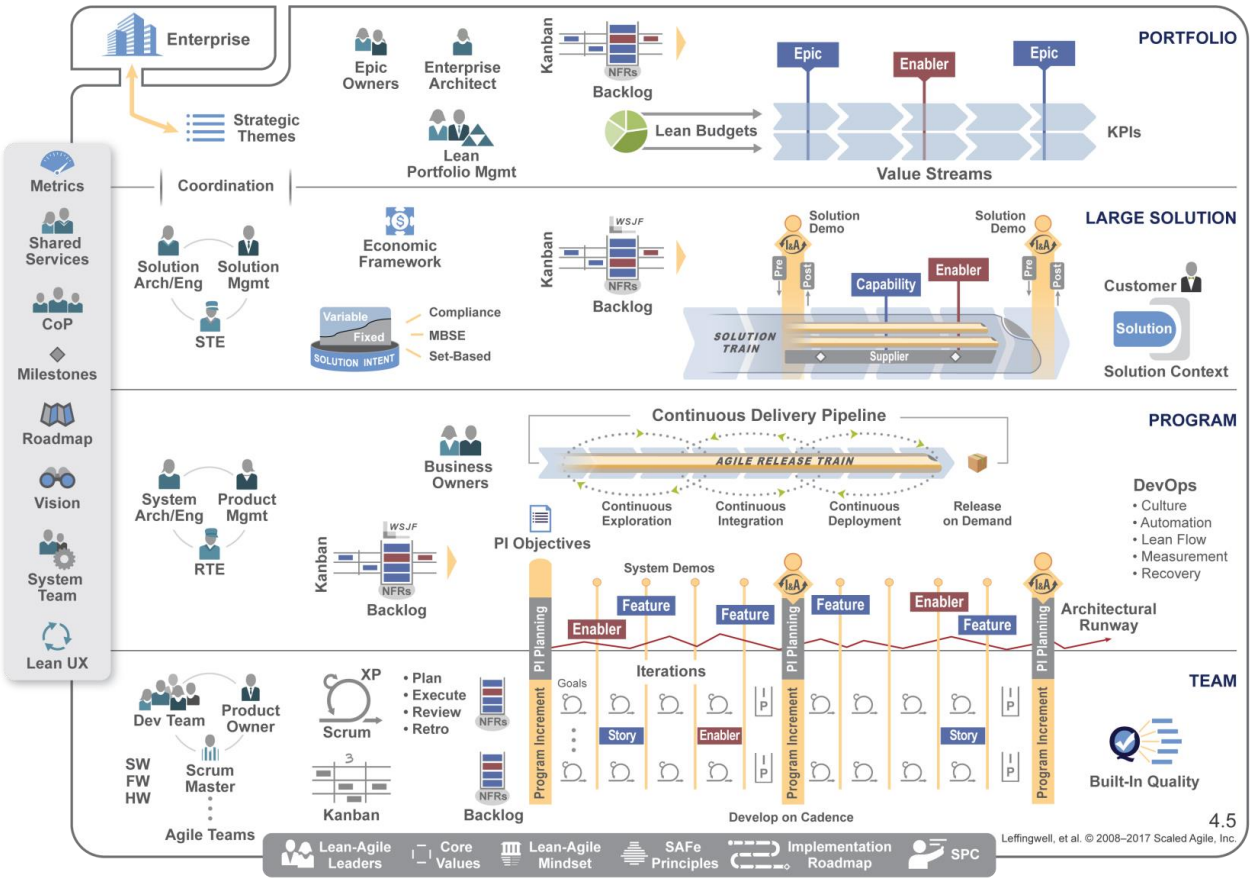
Knowledge for people building the world's most important systems

SAFe® is a freely revealed knowledge base of integrated, proven patterns for enterprise Lean-Agile development.

scaledagileframework.com

The Scaled Agile Framework® (SAFe®)

Synchronizes alignment, collaboration, and delivery for large numbers of teams.



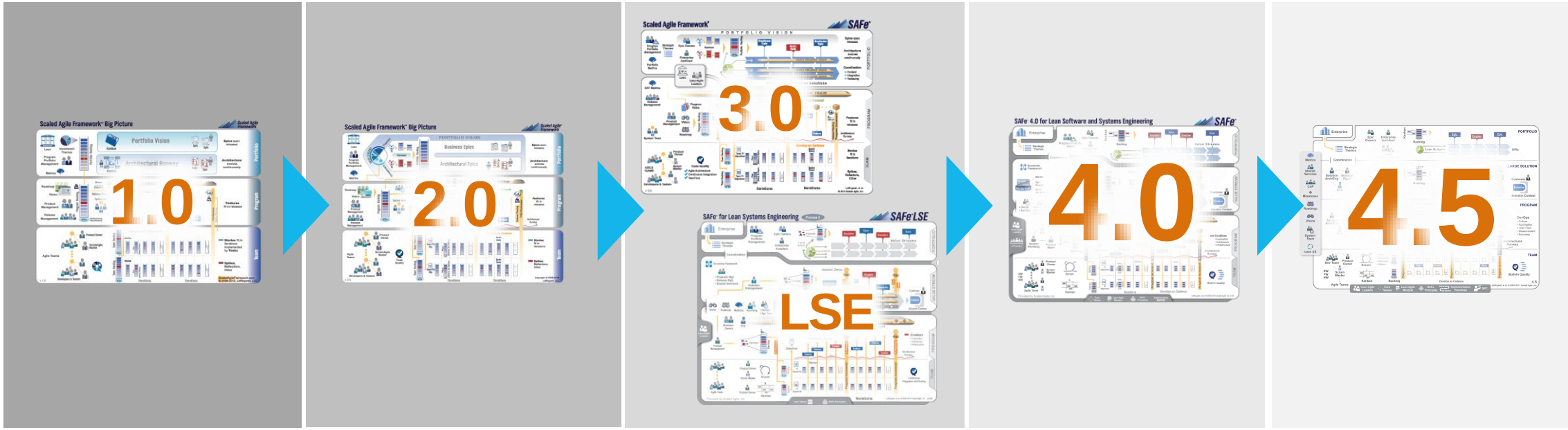
Core Values

1. Built-In Quality
2. Program execution
3. Alignment
4. Transparency

Roots, past, present and future

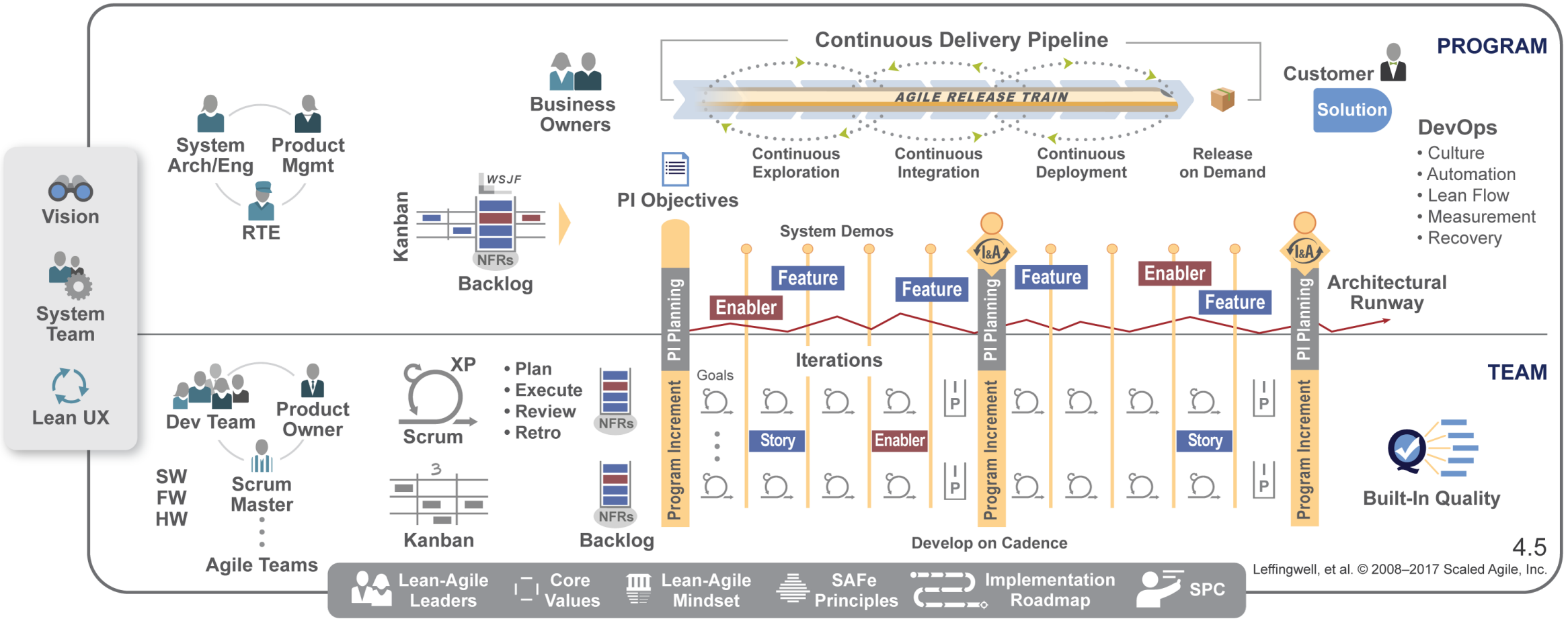
Field experience at enterprise scale

2011 — Now…



Agile development | Lean product development | Systems thinking

Essential SAFe provides the basis for success



4.5

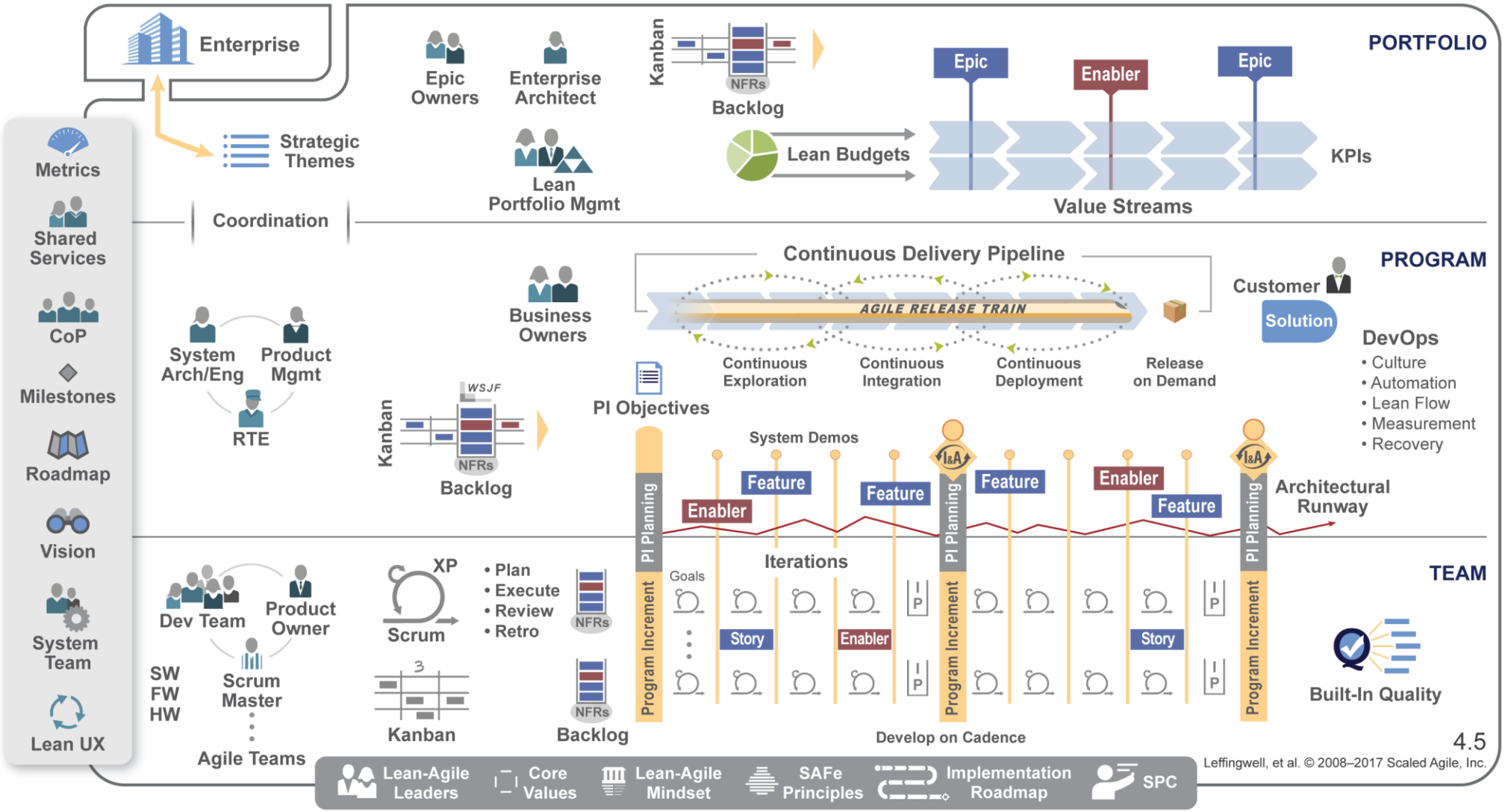
Nothing beats an Agile Team

- Empowered, self-organizing, self-managing, cross-functional team
- Delivers valuable, tested, working system every two weeks
- Uses a team framework which combines the best of Scrum project management, XP-inspired technical practices and Kanban for flow

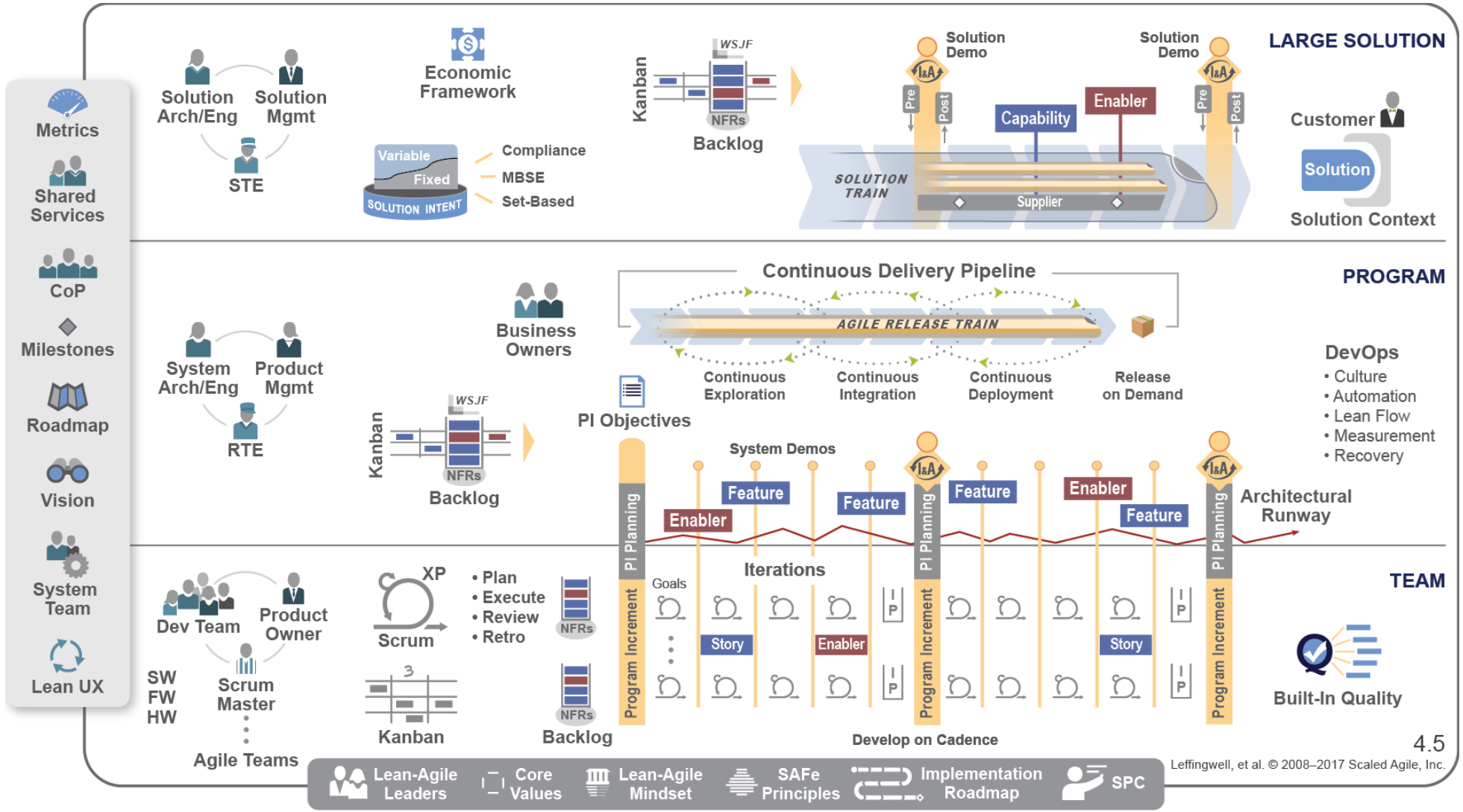
Except a team of Agile Teams

- Self-organizing, self-managing, team of agile teams
- Delivers working, tested full system increments every two weeks
- Operates with Vision, architecture and UX guidance
- Common iteration lengths and estimating
- Face-to-face planning for collaboration, alignment, and adaptation

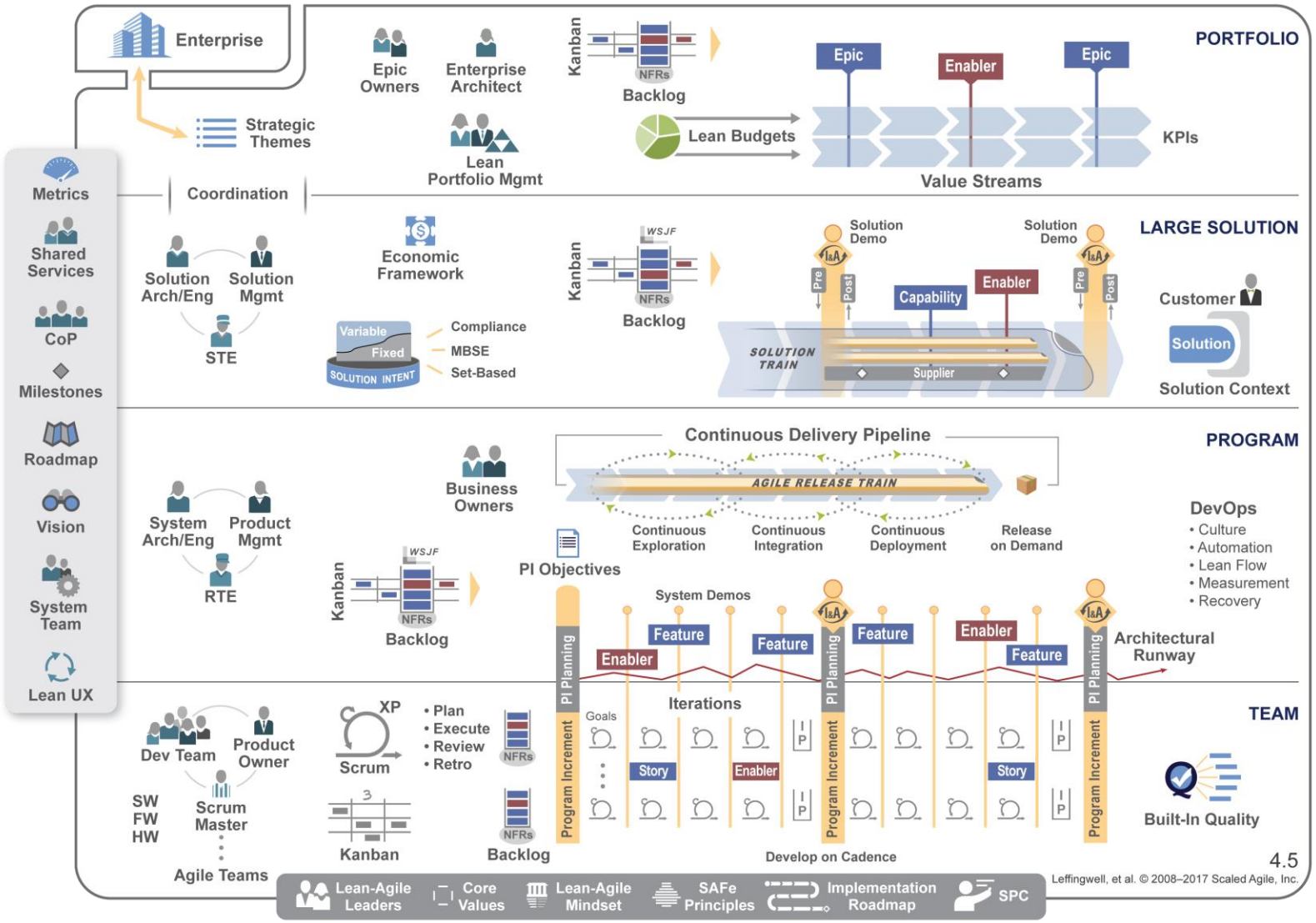
Portfolio SAFe adds Lean Portfolio governance



Large Solution SAFe is coordinated by a Solution Train



Some enterprises require Full SAFe



Get business results

10 – 50% happier, more motivated employees

30 – 75% faster time-to-market

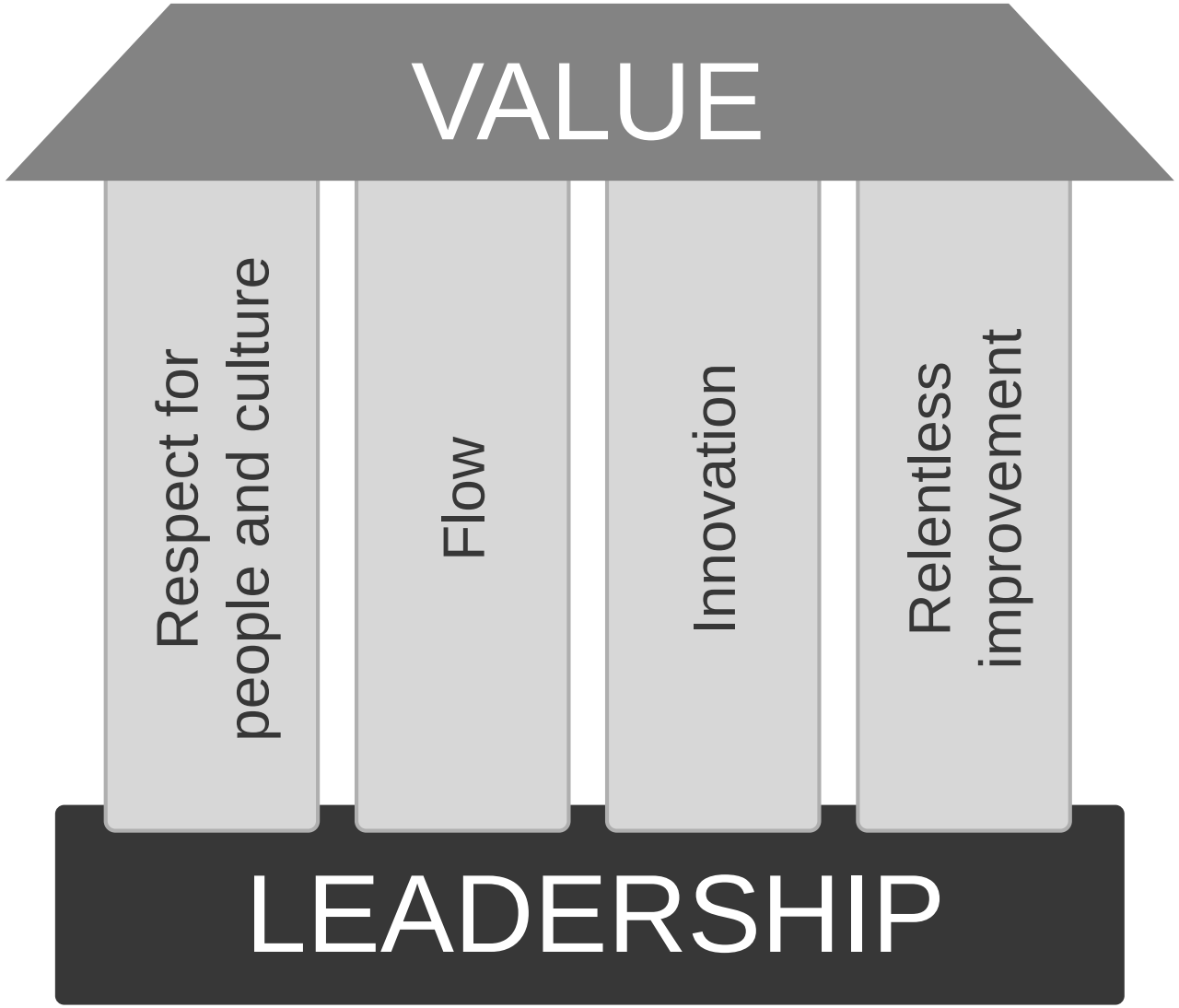
20 – 50% increase in productivity

25 – 75% defect reduction

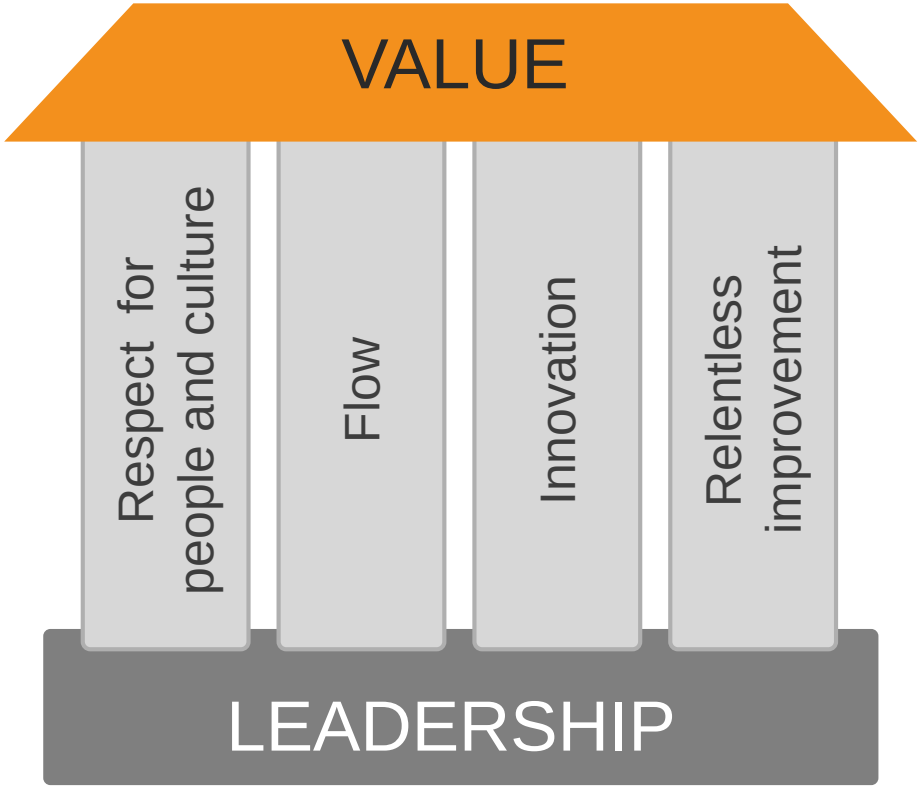
ScaledAgileFramework.com/**case-studies**

1.2 Explore Lean, the Agile Manifesto, and SAFe Principles

SAFe House of Lean



Purpose

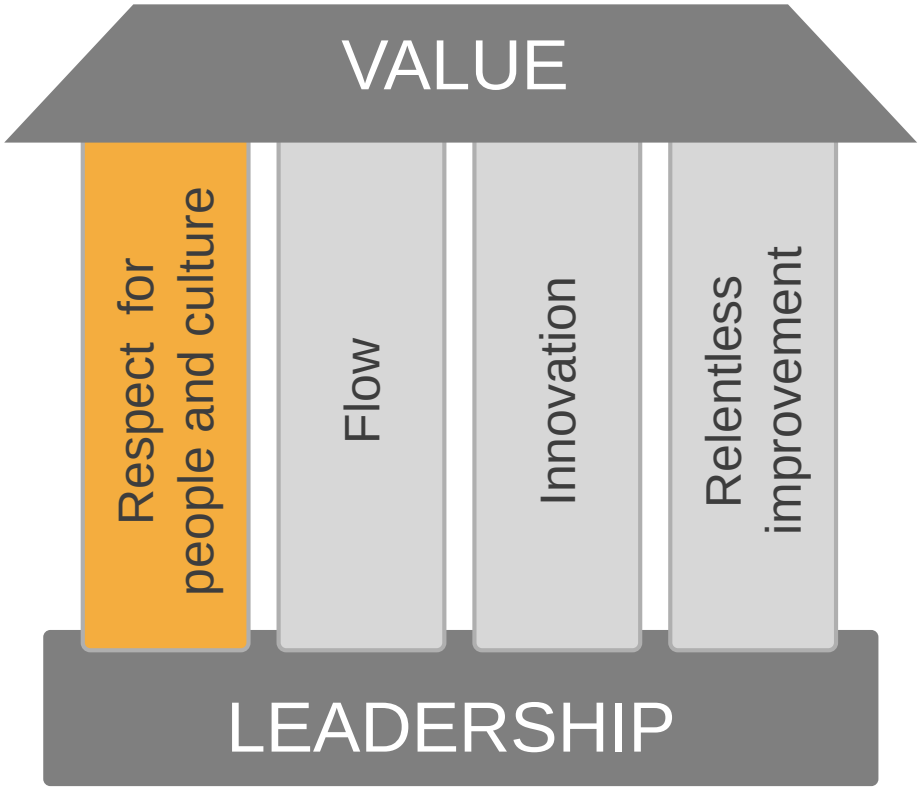


Achieve the sustainably shortest lead time with:

- Best quality and value to people and society
- High morale, safety and customer delight

There is only one boss. The customer.
And he can fire everybody in the company.
—Sam Walton

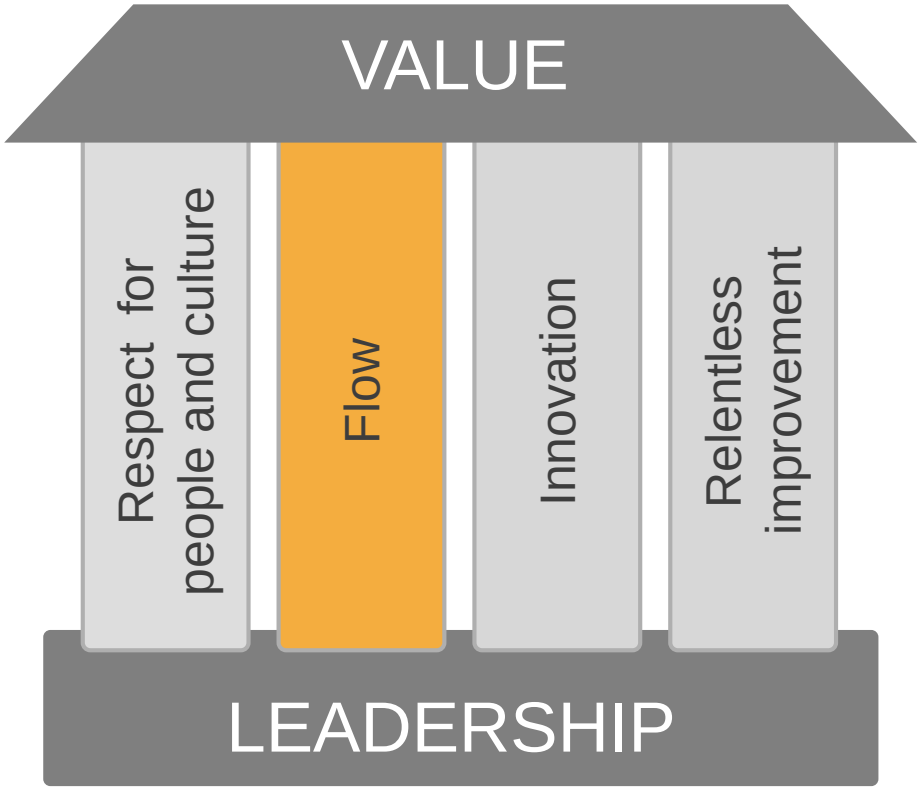
Respect for people and culture



Culture eats strategy for breakfast.
—Peter Drucker

- People do all the work
- Your customer is whoever consumes your work
  - Don’t overload them
  - Don’t make them wait
  - Don’t force them to do wasteful work
  - Don’t impose wishful thinking
- Build long-term partnerships based on trust
- Cultural change comes last, not first
- To change the culture, you have to change the organization

Flow

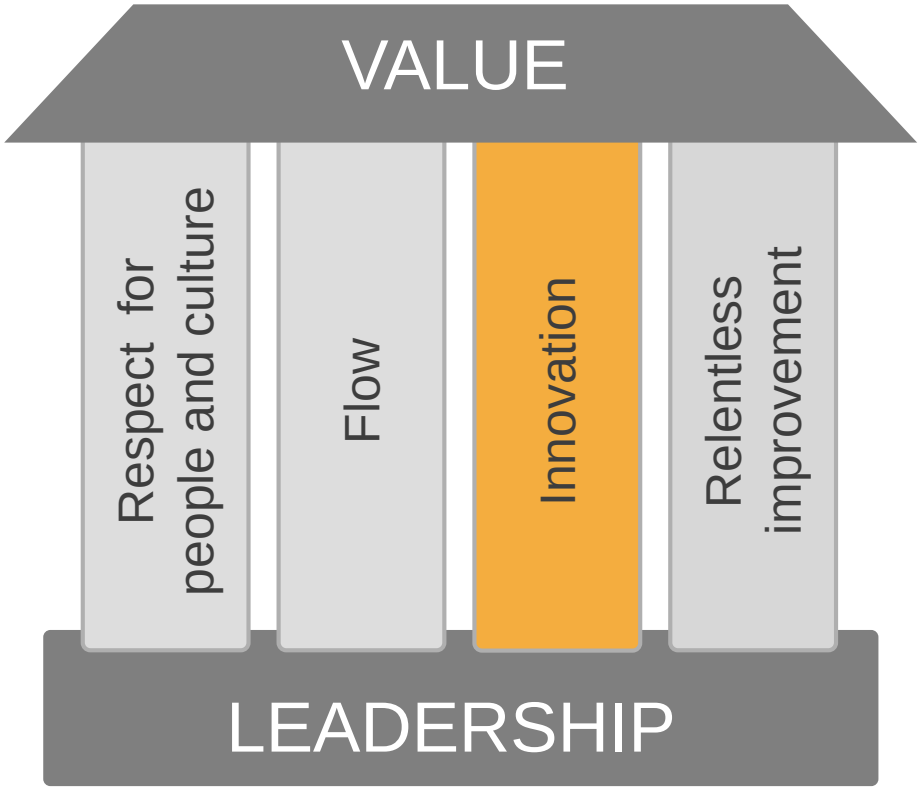


Operating a product development process near full utilization is an economic disaster.

—Don Reinertsen

- Optimize continuous and sustainable throughput of value
- Avoid start-stop-start project delays
- Build quality in; flow depends on it
- Understand, exploit and manage variability
- Integrate frequently
- Informed decision-making via fast feedback

Innovation



Innovation comes from the producer.

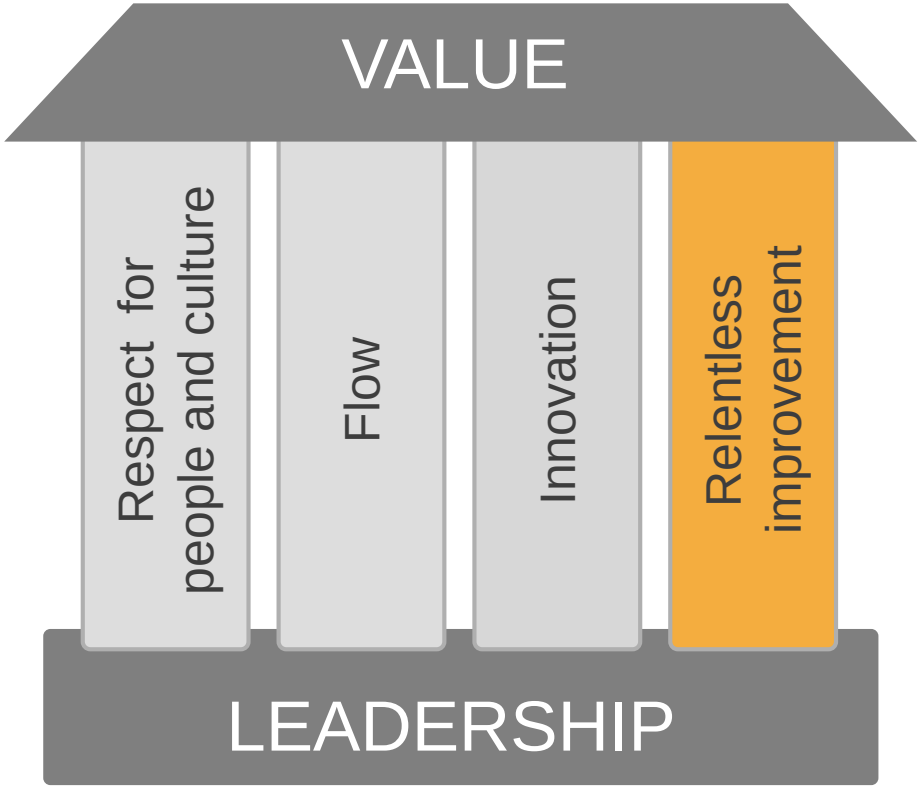
—W. Edwards Deming

- Producers innovate; customers validate
- Get out of the office (Gemba*)
  - *No useful improvement was ever invented at a desk*

    — Taiichi Ohno
- Provide time and space for creativity
- Apply innovation accounting
- Pivot without mercy or guilt

** Gemba: The "real place" where the work is actually done.*

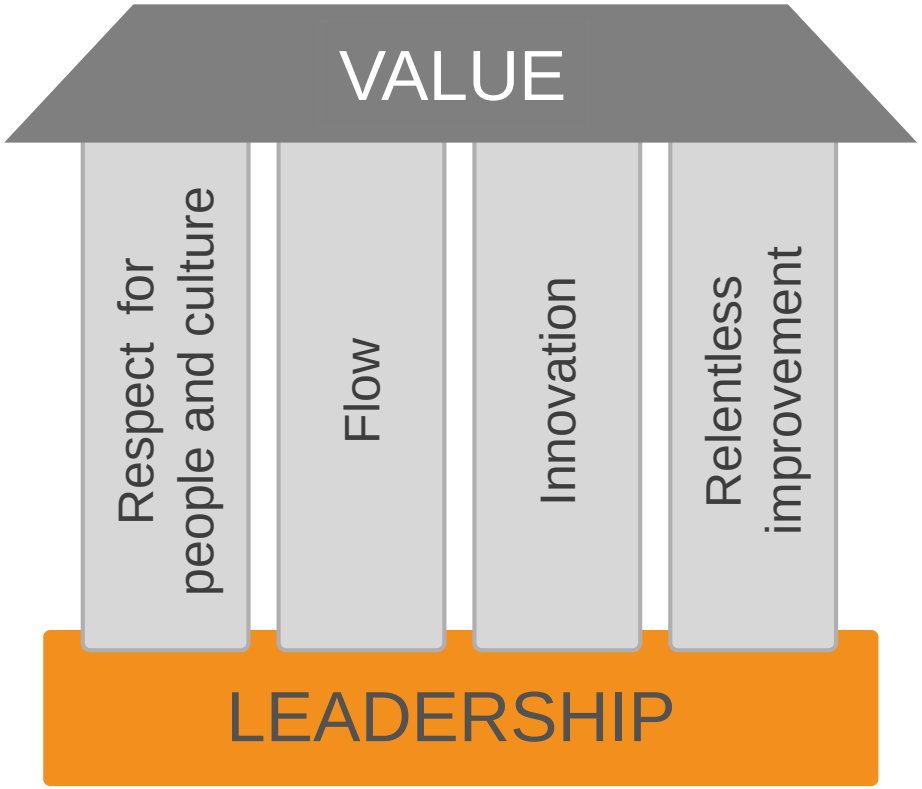
Relentless improvement



Those who adapt the fastest, win.

- A constant sense of danger
- Optimize the whole
- Consider facts carefully, then act quickly
- Apply lean tools to identify and address root causes
- Reflect at key milestones; identify and address shortcomings

Leadership



People are already doing their best; the problems are with the system. Only management can change the system.

—W. Edwards Deming

- Lead the change
- Know the way; emphasize life-long learning
- Develop people
- Inspire and align with mission; minimize constraints
- Decentralize decision-making
- Unlock the intrinsic motivation of knowledge workers

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

Agile Manifesto

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

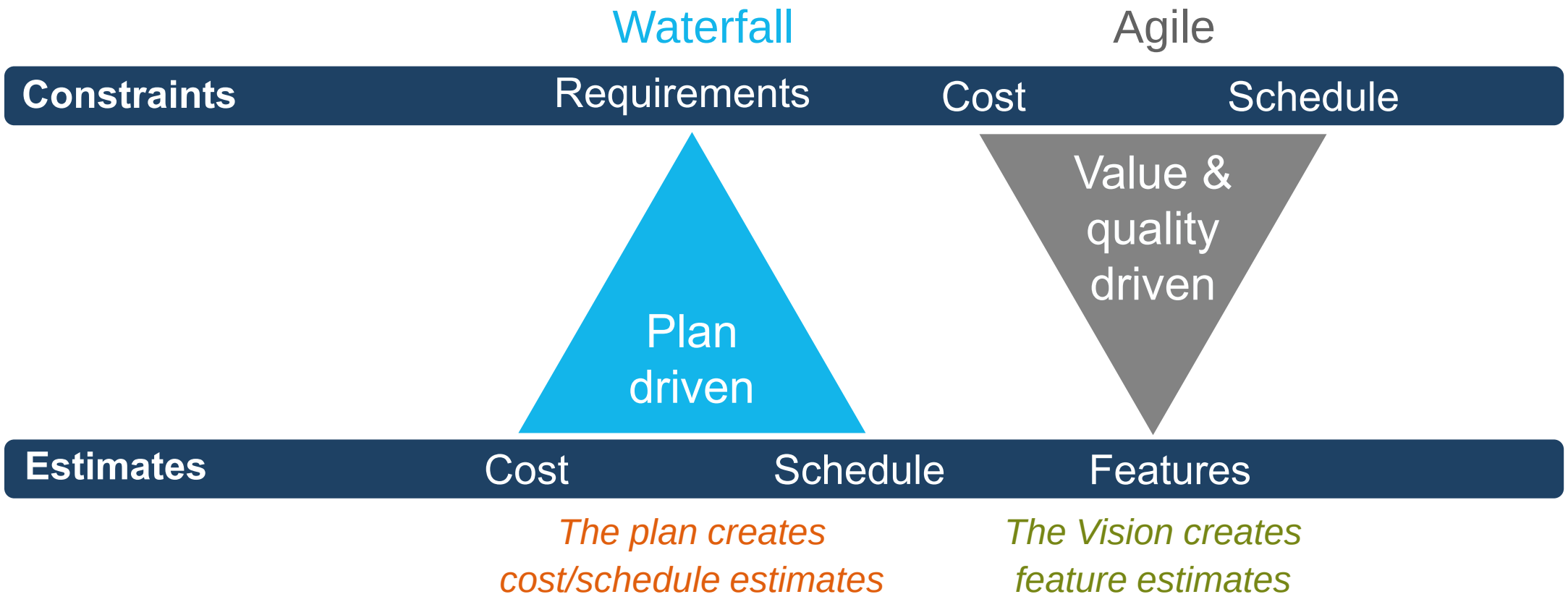
agilemanifesto.org/principles.html

Agile Manifesto

7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

agilemanifesto.org/principles.html

Agile turns development upside-down



- Agile Teams show that *dates* matter and they *meet* their commitments
- Business Owners understand how *priorities* matter
- Fix *quality*, not scope

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

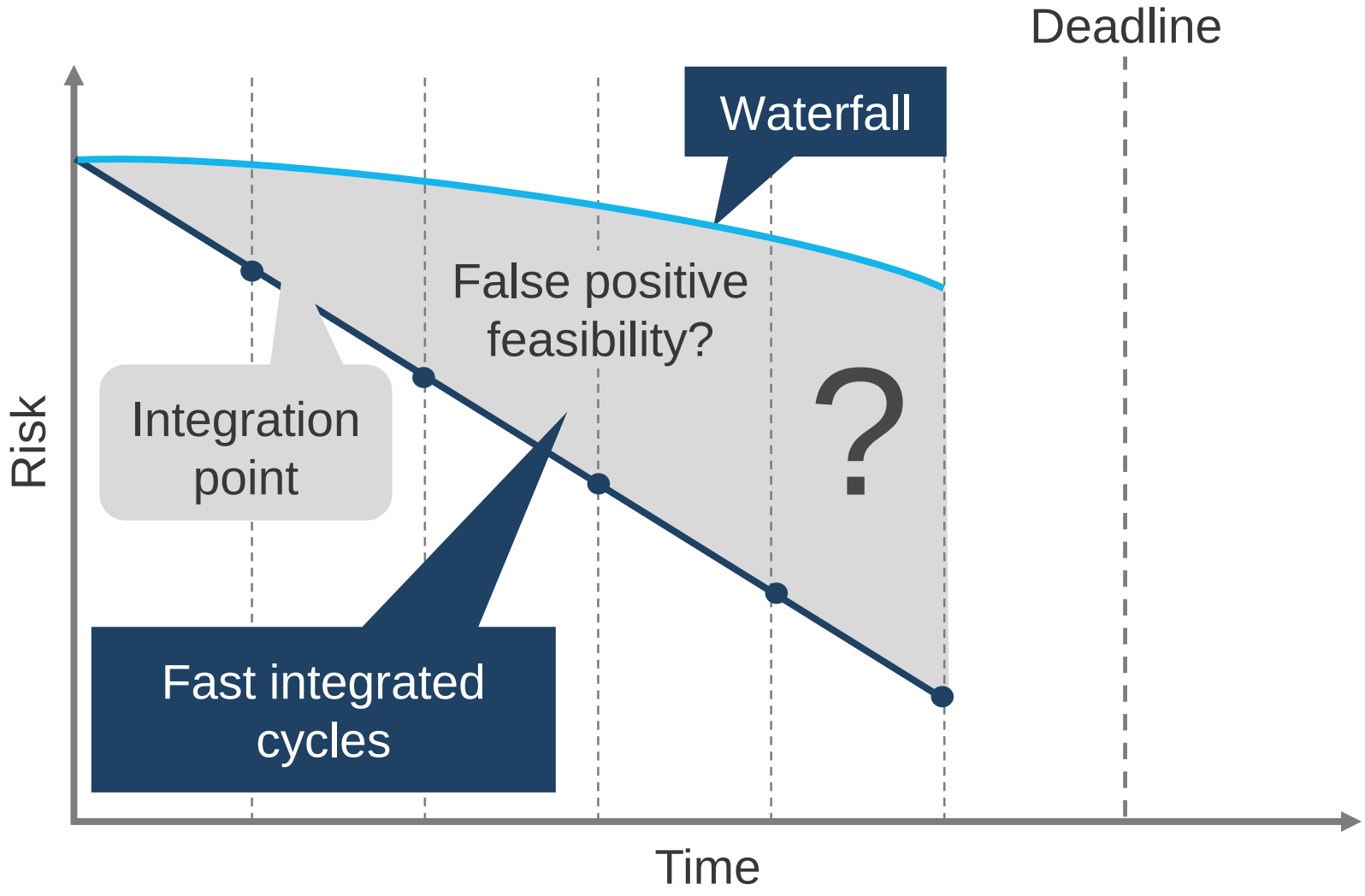
#6-Visualize and limit WIP, reduce batch sizes, and manage queue lengths

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

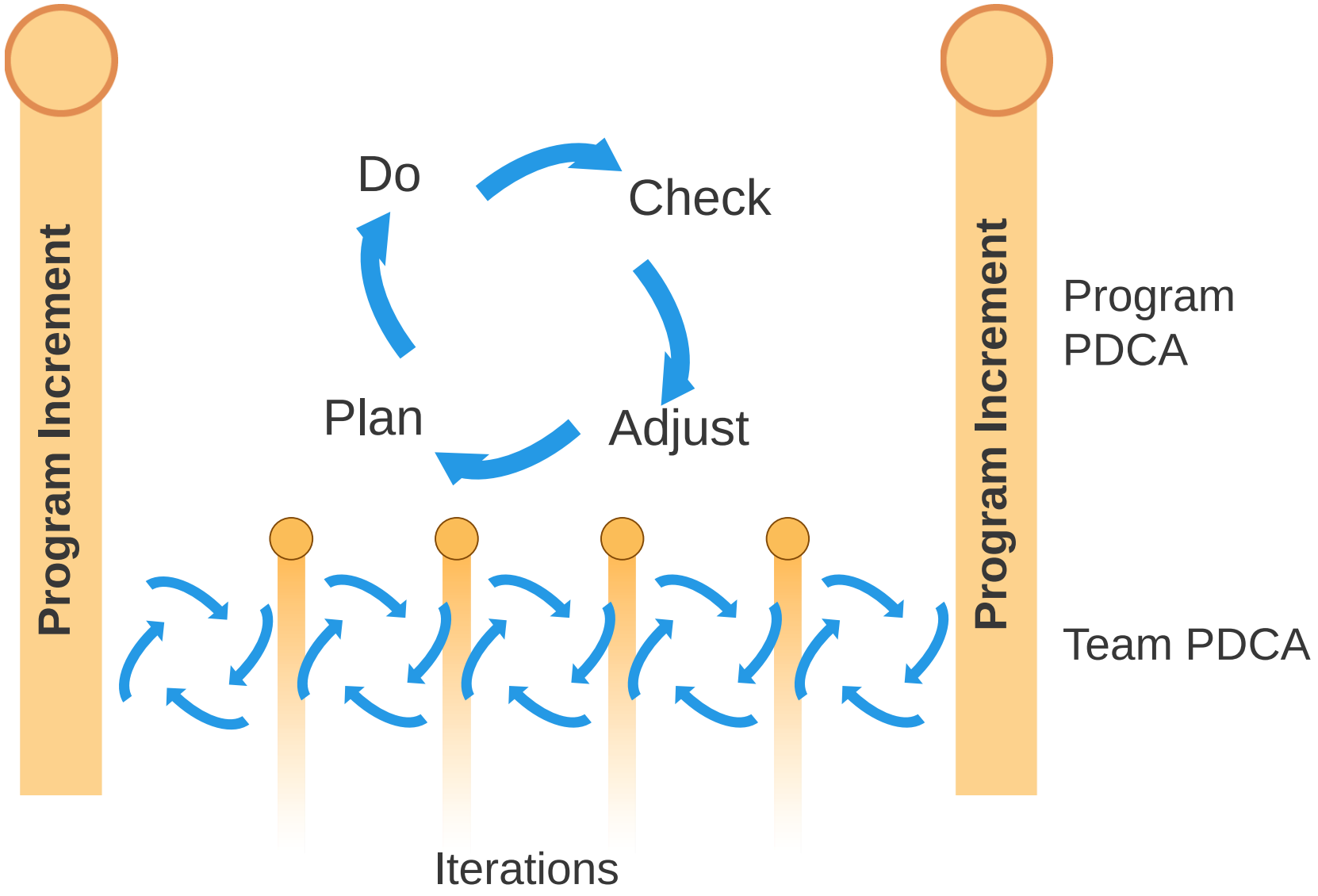
#8-Unlock the intrinsic motivation of knowledge workers

#9-Decentralize decision-making

#4 ild incrementally with fast, integrated learning cycles



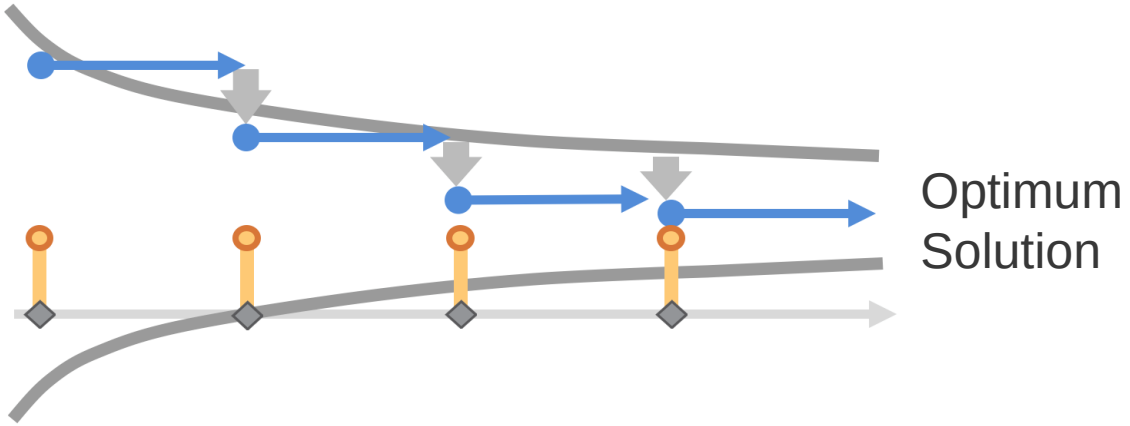
Use Iterations and Program Increments to learn fast



#5 Base milestones on objective evaluation of working systems

- Phase gates force too-early design decisions, encourage false-positive feasibility
- Assume a "point" Solution exists and can be built right the first time

Objective Milestones facilitate learning and allow for continuous, cost-effective adjustments toward an optimum Solution

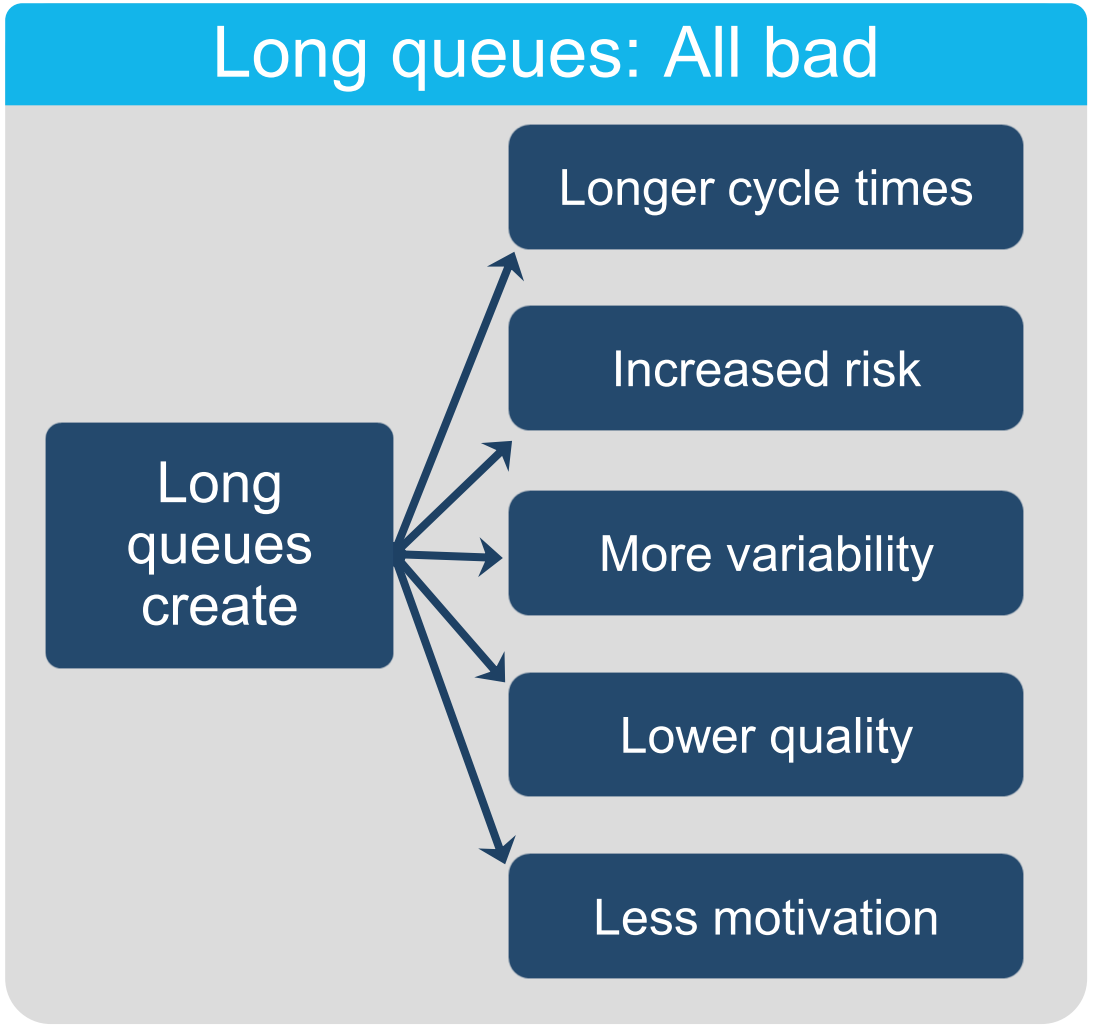


#6 Visualize and limit WIP, reduce batch size, and manage queue lengths

- Understand Little's Law
- Faster processing time decreases wait
- Control wait times by controlling queue lengths

$$W_q = \frac{L_q}{\lambda}$$

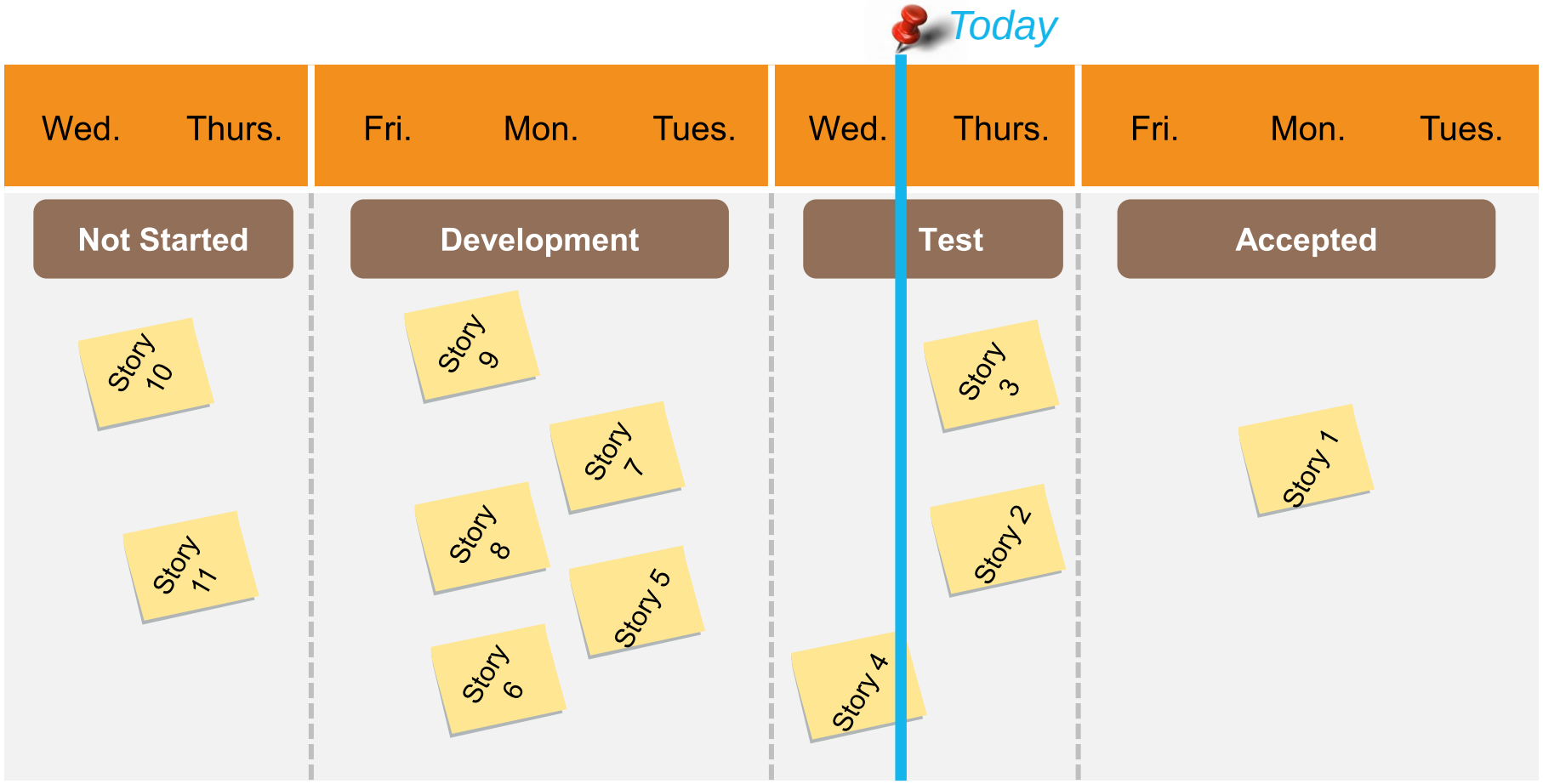
Average wait time = average queue length / average processing rate



Principles of Product Development Flow, Don Reinertsen

Visualize and limit work in progress

One team's Big Visible Information Radiator (BVIR)



How is this team doing? How do you know that?

Exercise: Large batch push

- In your team, choose a four-person group who will process the 10 coins on the table. One additional person is the group timekeeper. Other members are individual timekeepers.
- Each of the four people flips all coins one at a time, recording his own results (heads or tails)
- Then each person passes all coins at the same time to the next person
- The timekeeper records time from the start of the first flip to the completion of the last flip for the group. Each individual timekeeper records time for a single individual.

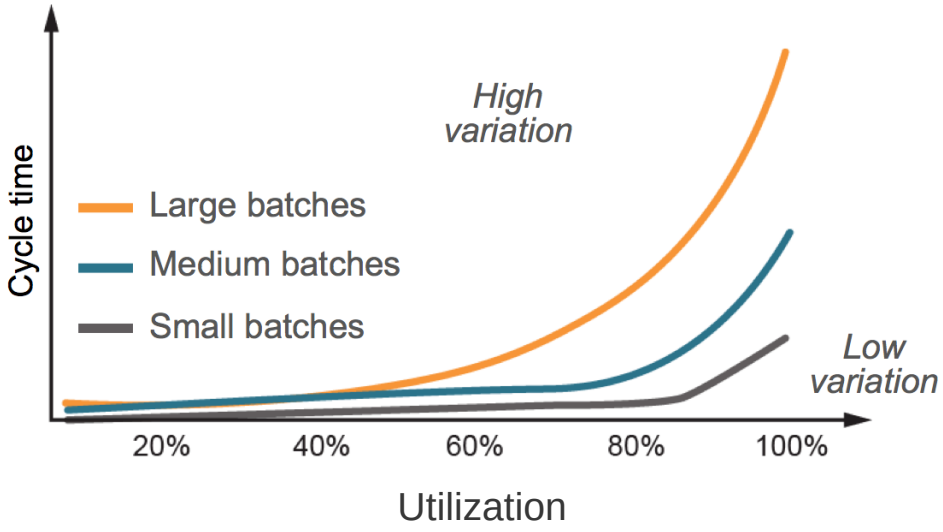
Exercise: Small batch pull

- Similar four-person process
- Each of the four people flips each coin one at a time and records the result
- People pull coins from the previous person as soon as he is done recording them, and process them immediately
- The timekeeper records the time from the start of the first flip to the completion of the last flip for the group. Each individual timekeeper records time for a single individual.

Reduce batch size

Small batches go through the system faster, with lower variability.

- Large batch sizes increase variability
- High utilization increases variability
- Severe project slippage is the most likely result
- Most important batch is the transport (handoff) batch
- Proximity (co-location) enables small batch size
- Good infrastructure enables small batches

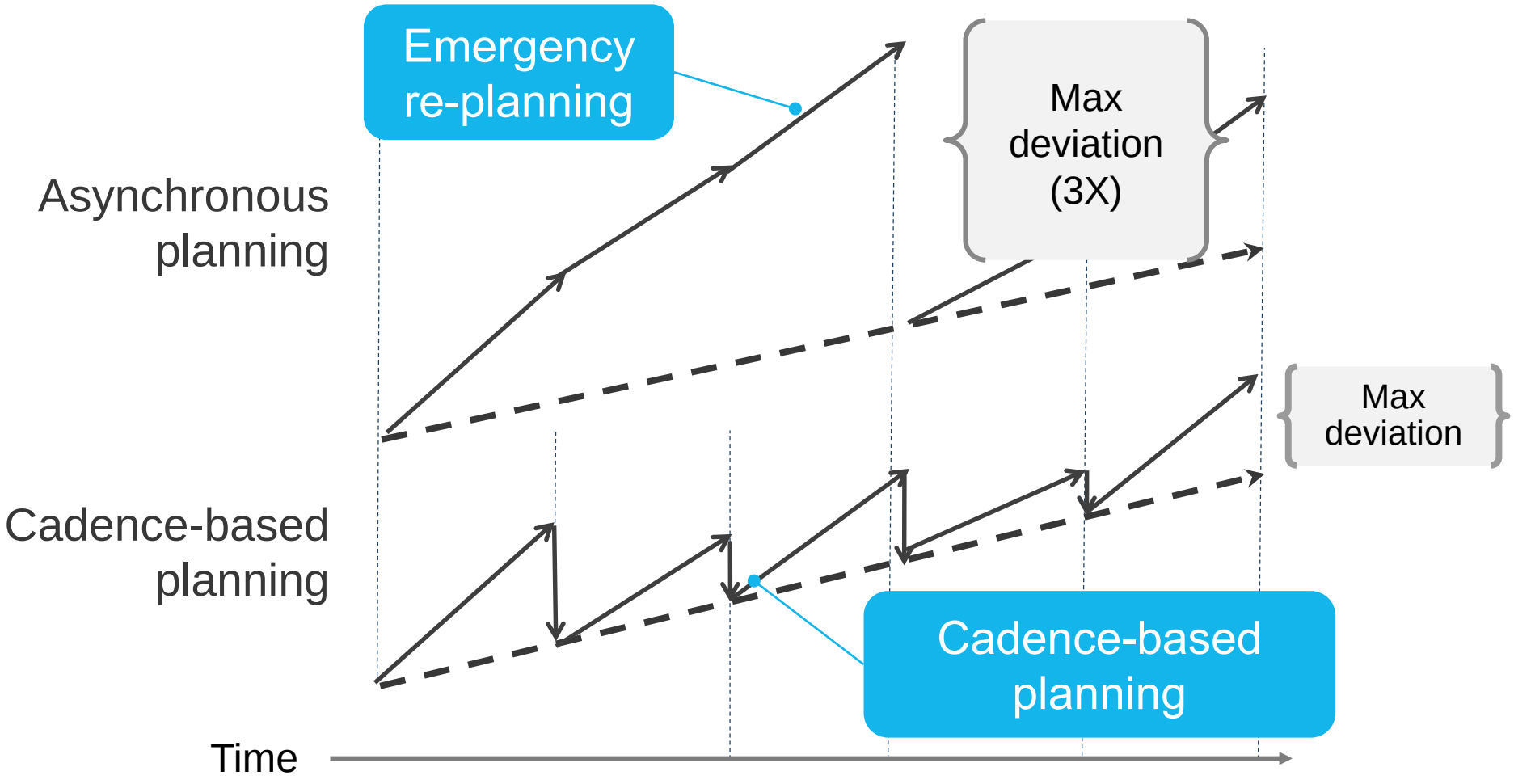


Implementing Lean Software Development,
Mary Poppendieck

Principles of Product Development Flow,
Don Reinertsen

#7 ntrol variability with planning cadence

Cadence-based planning limits variability to a single interval.



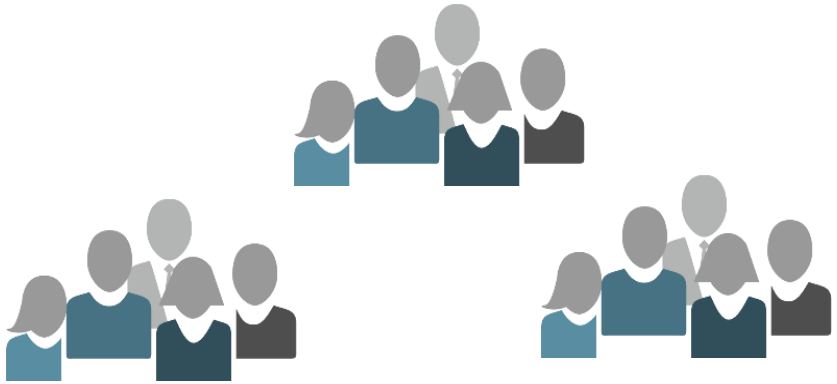
1.3 Identify Scrum, Kanban, and XP Practices

From traditional development to Agile

Instead of a large group

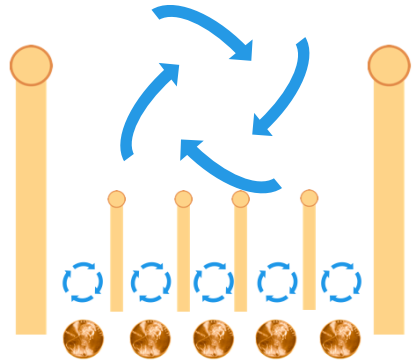
Working on all the requirements

Integrating and delivering value toward the end of development



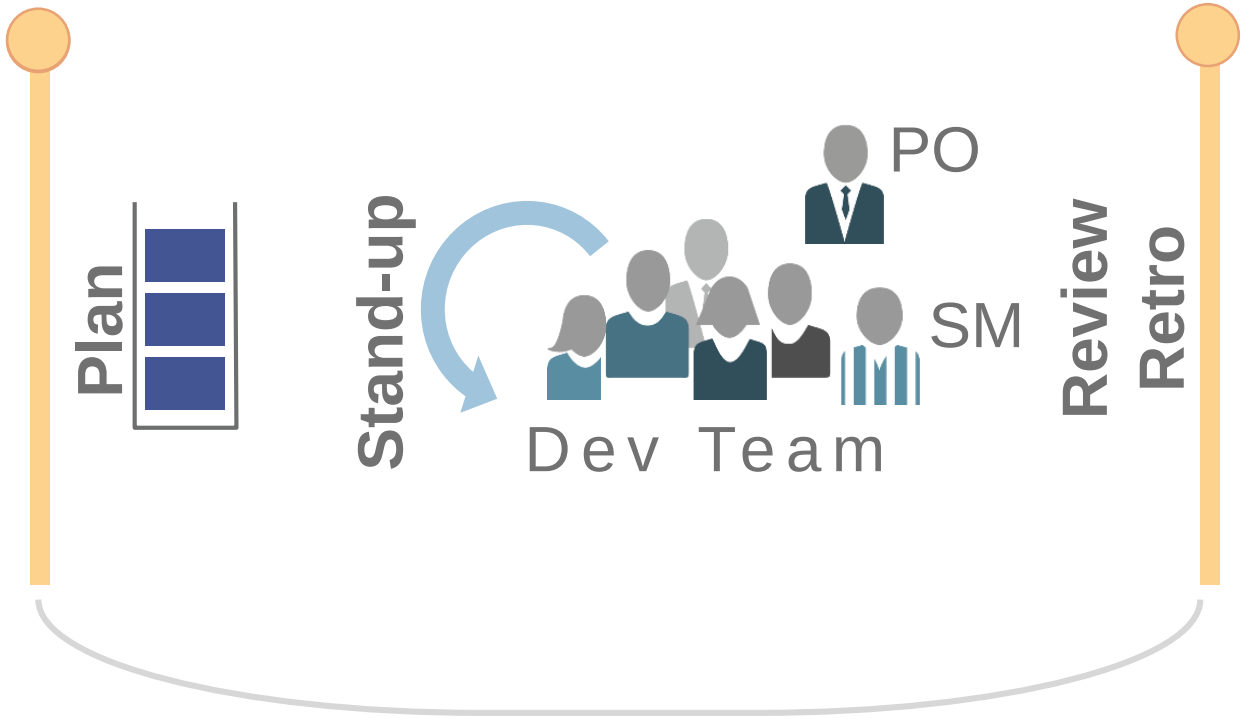
Have small teams working together as a program

Working on small batches of requirements



Delivering value in short timeboxes with frequent integration and improvement cycles

Agile for Teams: Scrum



Three roles

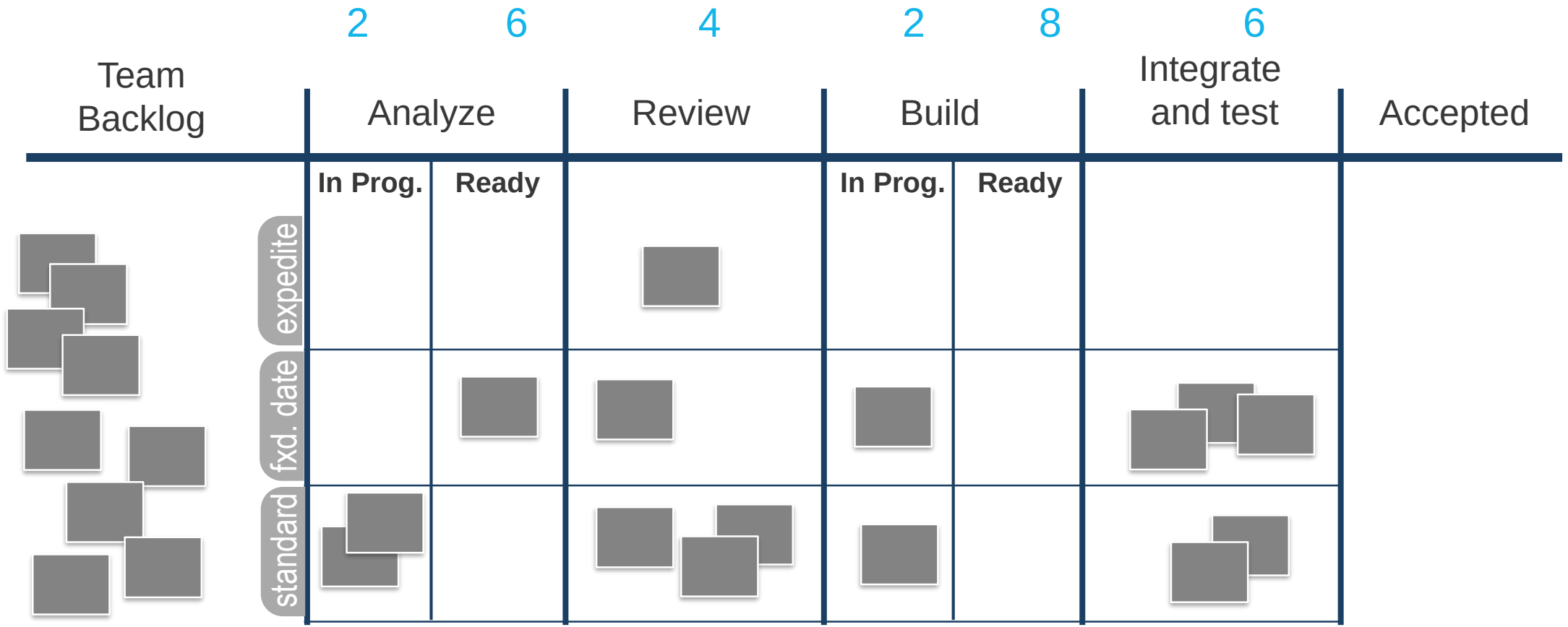
1. Development Team
2. Scrum Master (SM)
3. Product Owner (PO)

Four meetings

1. Iteration Planning (Lesson 3)
2. Daily Stand-up (Lesson 4)
3. Iteration Review (Lesson 4)
4. Iteration Retrospective (Lesson 4)

Agile for Teams: Kanban

Visualize work flow. Limit work in process. Improve flow.



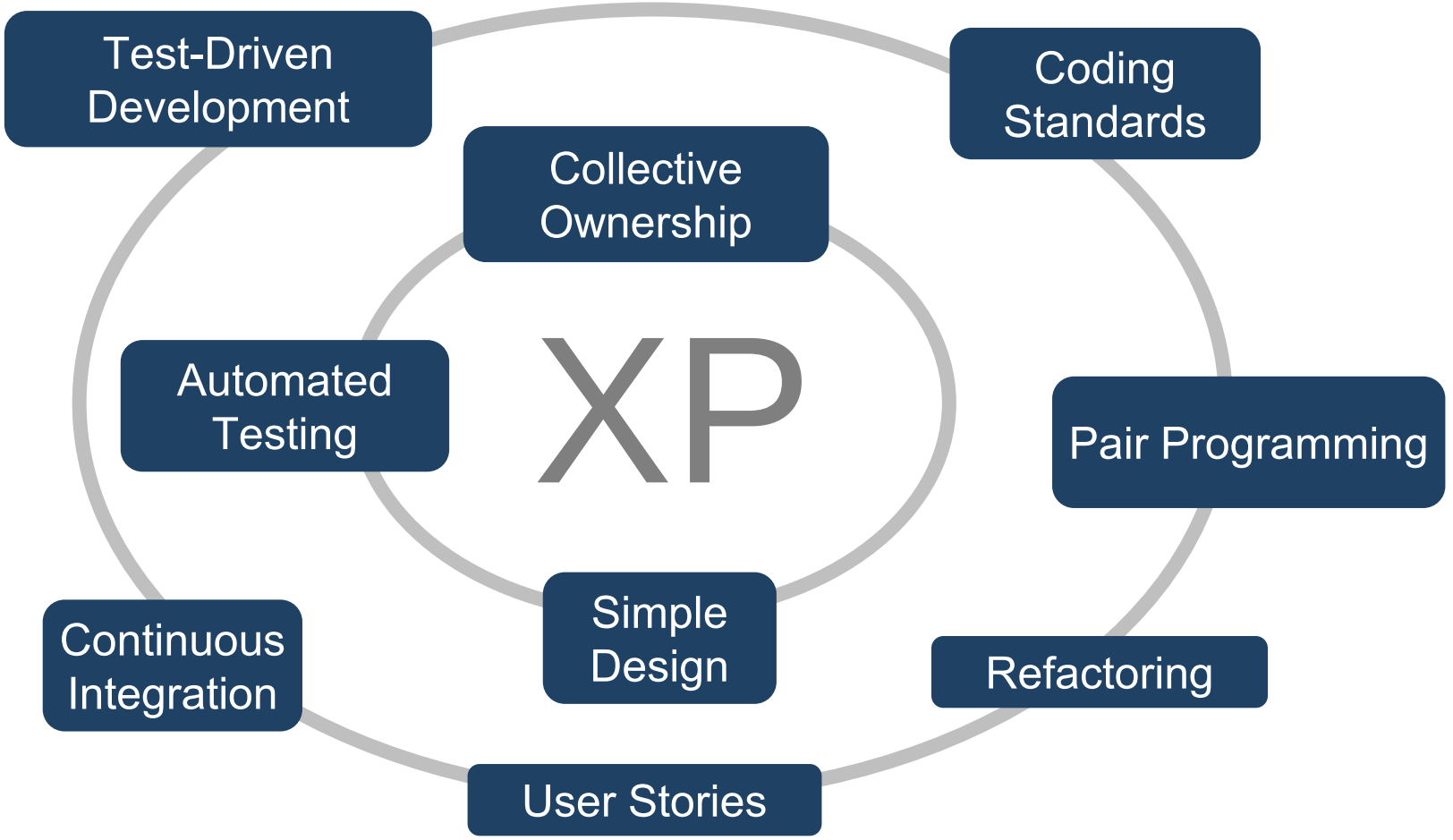
Agile for Teams: XP

XP practices drive endemic code quality to unprecedented levels.

Most high-performance teams use Scrum and XP together.

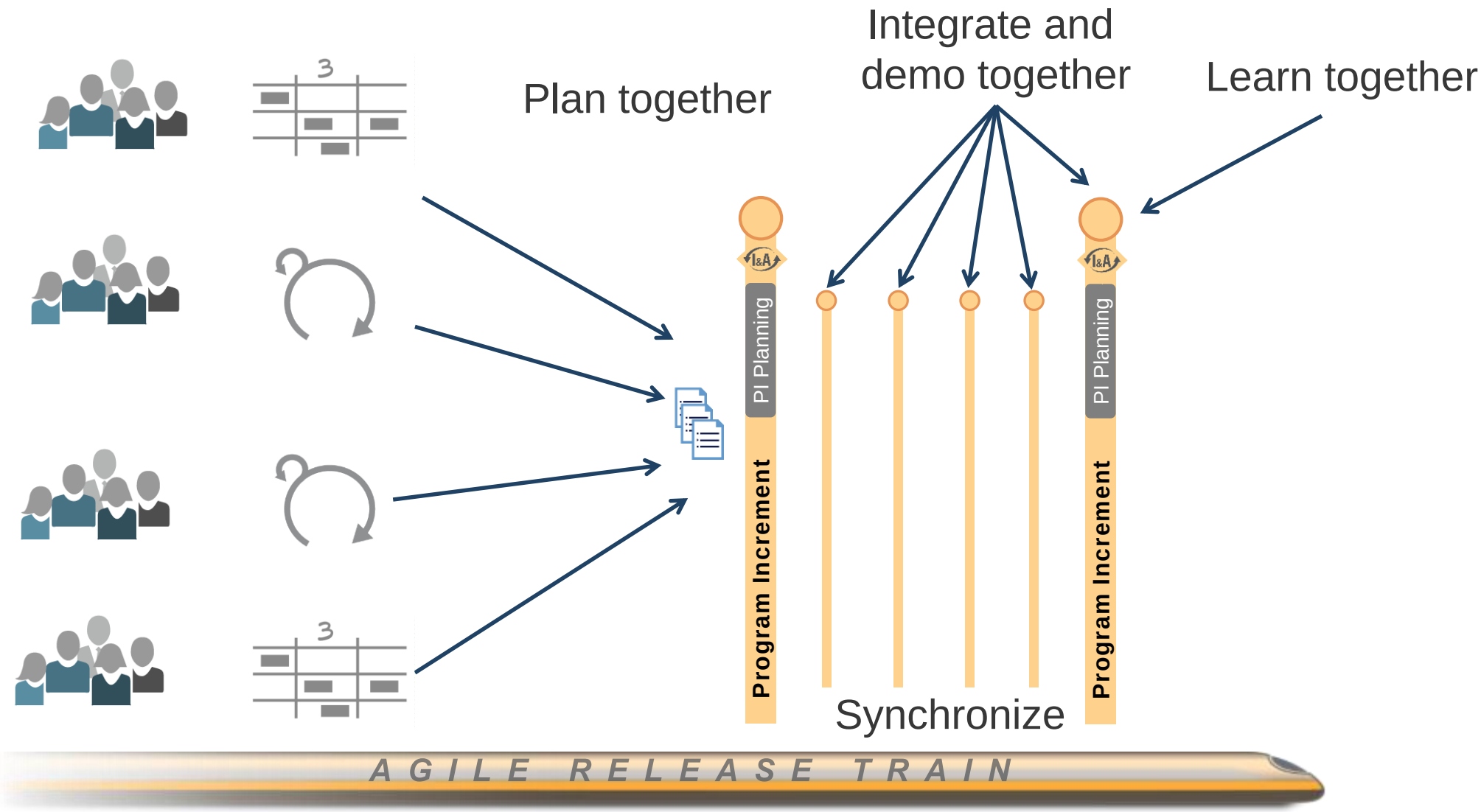
It is hard to get a Scrum with extreme velocity without XP engineering practices.

—Jeff Sutherland, co-creator of Scrum

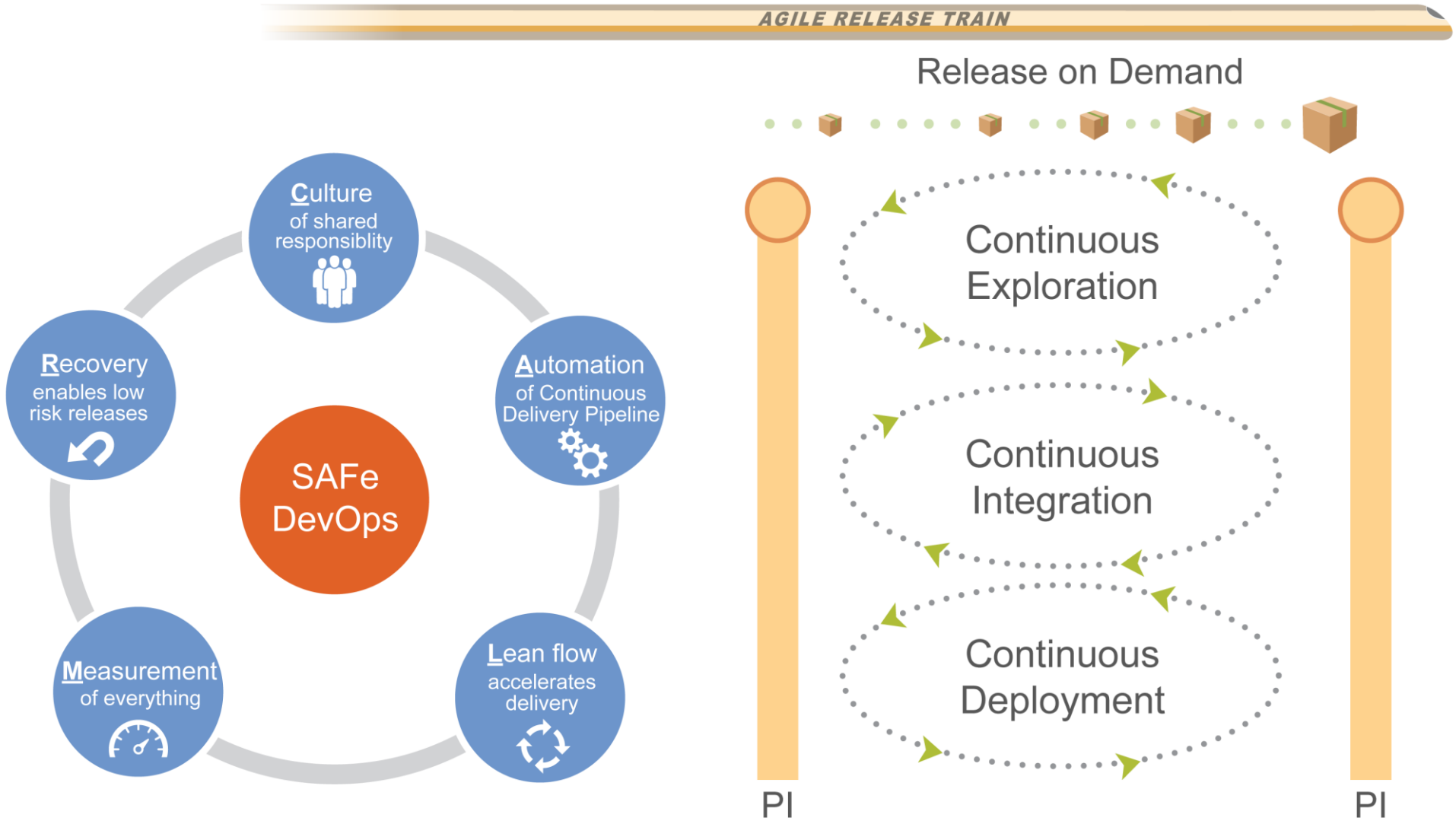


Adapted from xprogramming.com

Teams in SAFe are part of an Agile Release Train



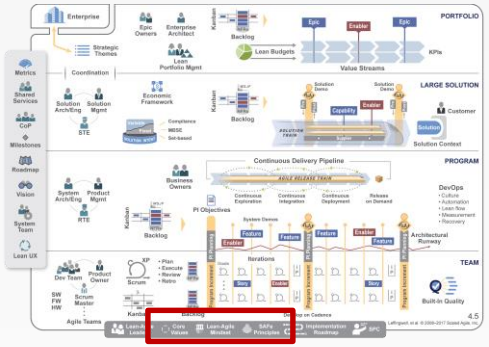
The ART and Teams continuously deliver value



Lesson summary

In this lesson, you:

- Connected with SAFe
- Explored Lean, the Agile Manifesto, and SAFe Principles
- Identified Scrum, Kanban, and XP practices



Suggested Scaled Agile Framework reading:
"Core Values," "Lean-Agile Mindset, and "SAFe Principles" articles

Lesson 2
Building an Agile Team

1. Introducing the Scaled Agile Framework
2. Building an Agile Team
3. Planning the Iteration
4. Executing the Iteration
5. Executing the PI

SAFe® Course Attending this course gives students access to the SAFe Practitioner exam and related preparation materials.

Learning objectives

2.1 Build your team

2.2 Explore the Scrum Master and Product Owner roles

2.3 Meet the teams and people on the train

2.1 Build your team

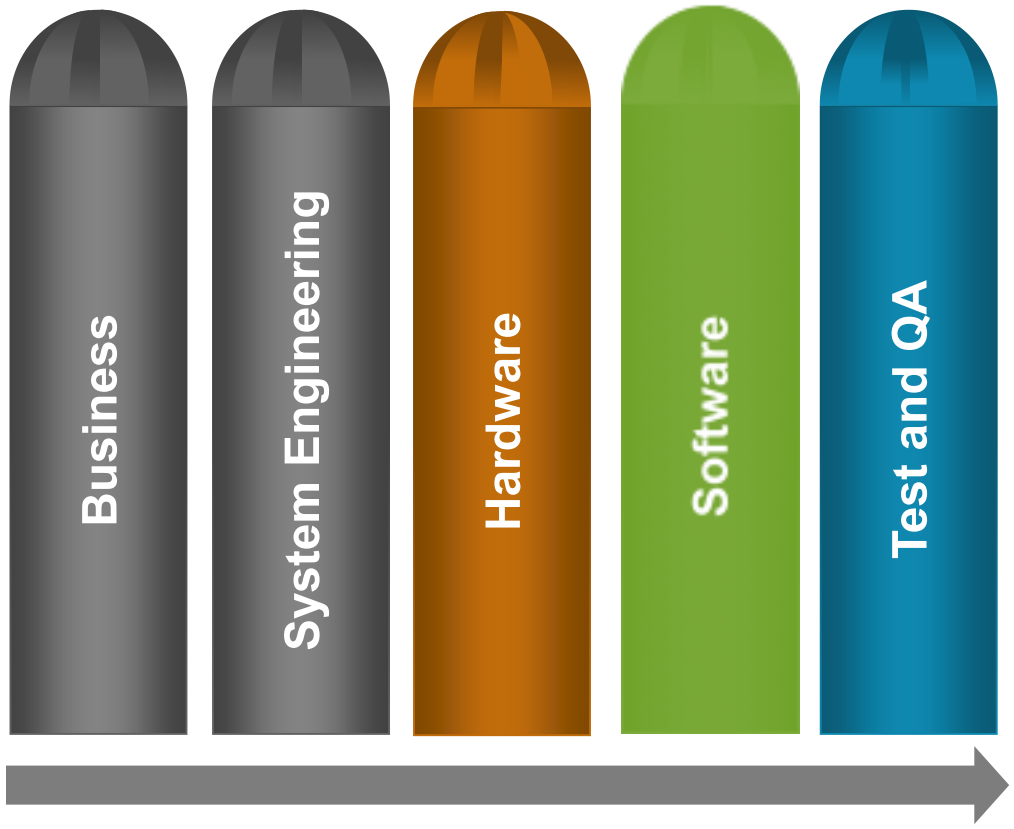
The Agile Team

*The 'relay race' approach to product development … may conflict with the goals of maximum speed and flexibility. Instead, a holistic or **'rugby'** approach—where a team tries to go the distance as a unit, passing the ball back and forth—may better serve today's competitive requirements.*

Youtu.be/KWnwI0-aeq0

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

Value doesn't follow silos

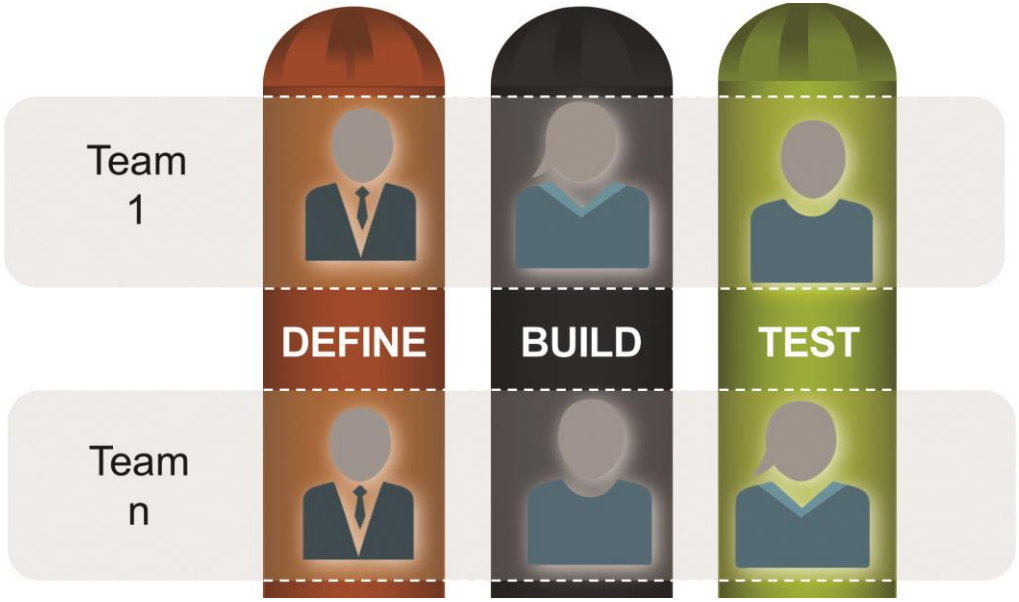


Management challenge:
Connect the silos

- Optimized for vertical communication
- Friction across the silos
- Location via function
- Political boundaries between functions

Build cross-functional Agile Teams

- Cross-functional, self-organizing entities that can define, build, and test a feature or component
- Optimized for communication and delivery of value
- Deliver value every two weeks



The power of “ba”

We, the work, and the knowledge are all one

- Dynamic interaction of individuals and the organization in the form of a self-organizing team; the fuel of “ba” is its self-organizing nature
- Team members create new points of view and resolve contradictions through dialogue
- “Ba” is energized with intentions, vision, interest, and mission
- Leaders provide autonomy, variety, trust, and commitment
- There is creative chaos via demanding performance goals
- The team is challenged to question every norm of development
- Equal access to information at all levels is critical

youtu.be/yiKFYTFJ_kw

Nishida, 1921, 1970; *Hitotsubashi on Knowledge Management,* Takeuchi and Nonaka, 2004

Teamwork

It is amazing what you can accomplish if you do not care who gets the credit.

—Harry Truman

Exercise: Ball game

- Game objective is to process as many balls as possible as one team
- Balls must be passed with air time between any two people. They must be touched by each person. A ball must return to start point before it is counted complete.
- You will have 2 minutes to organize and begin your first Iteration
- You will do 5 iterations. Each iteration is 2 minutes, followed by a 1 minute retrospective.
- To get credit, you must provide an estimate for the number of balls you think you can process before each iteration

Collocated teams enhance productivity

Collocation:

- Critical for the Agile Team to be effective
- Recommended for programs to have efficient product development flow
- Distributed development must be compensated with efficient remote interaction (video-conferencing, sharing and collaboration tools, Agile project management tools, etc.)

Team working area

Shared team area

Agile Teams power the train

Development Team

- Creates and refines user stories and acceptance criteria
- Defines/builds/tests/delivers Stories
- Develops and commits to Team PI Objectives and Iteration plans

---

Product Owner

- Defines and accepts Stories
- Acts as the Customer for developer questions
- Works with Product Management to plan Releases
- A team has only one Product Owner, who may be dedicated to one or two teams

---

Scrum Master

- Runs team meetings, coaches Agile mindset and practices
- Removes impediments, protects the team from outside influence
- Attends Scrum of Scrum meetings
- May be a part-time role for a team member (25 – 50%), or a single Scrum Master may be shared across 2 – 3 teams

System Team

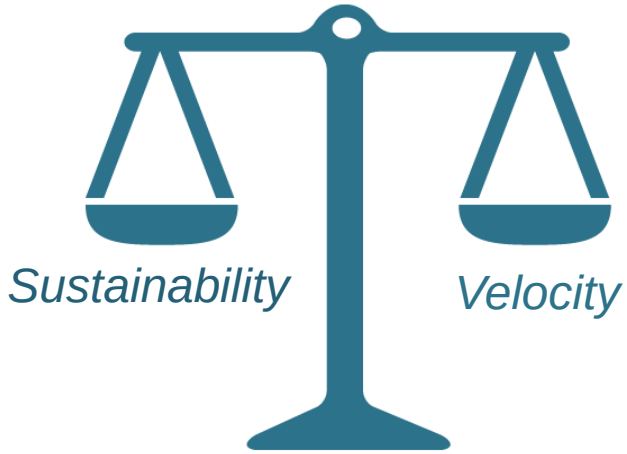
The System Team provides processes and tools to integrate and evaluate assets early and often.

- Builds the development infrastructure and manages environments
- Assists with test automation strategies and adoption
- Provides/supports full system integration
- Performs end-to-end system and performance testing
- Stages and supports the System Demos

Organizing teams around value

Organize for the larger purpose

- Maximize velocity by minimizing dependencies and handoffs, while sustaining architectural robustness and system qualities



A team can be organized around

- Features
- Components

Far less desirable

- Architectural layer
  - Platform, middleware, UI, DB, business logic, etc.
- Other
  - Programming language, spoken language, technology, location

Finding the right trade-off

Most large programs have a mix.

Lean toward **Feature Teams**:

- Fastest velocity
- Minimize dependencies
- Develop T-shaped skills

Use **Component Teams** when:

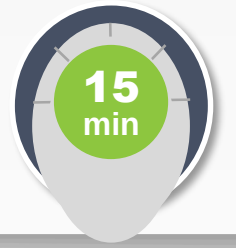
- High reuse, high technical specialization, critical NFRs
- Creating each component as a "potentially replaceable part of the system, with well-defined interfaces"

Generally avoid organizing around architectural layers, as they create team coupling and don't provide a technical separation of concerns.

Exercise: Build your team

As a **team**, you want to understand:

- What are your responsibilities and skill sets?
- Define our team name. Team names should not be the names of components, subsystems, or feature areas.
- Discuss your role as a feature or component team
- Discuss what your team is responsible for, and what other things you can do
- Choose a name for your team
- Prepare a short presentation about your team (name, role on the train, and special skills on the team that other teams should know about)



2.2 Explore the Scrum Master and Product Owner roles

Roles: Scrum Master

- Coaches the team
- Ensures that the team follows Agile principles and practices
- Facilitates processes and meetings
- Removes impediments and barriers
- Protects the team from external forces

The Scrum Master in the Enterprise

- Coordinates with other Scrum Masters, the System Team, and shared resources in the ART PI Planning meetings
- Works with the above teams throughout each Iteration and PI
- Coordinates with other Scrum Masters and the Release Train Engineer in Scrum of Scrums
- Fosters normalized estimating within the team
- Helps teams operate under architectural and portfolio governance, System Level integration, and System Demos
- Fosters adoption of Agile technical practices

Roles: Product Owner

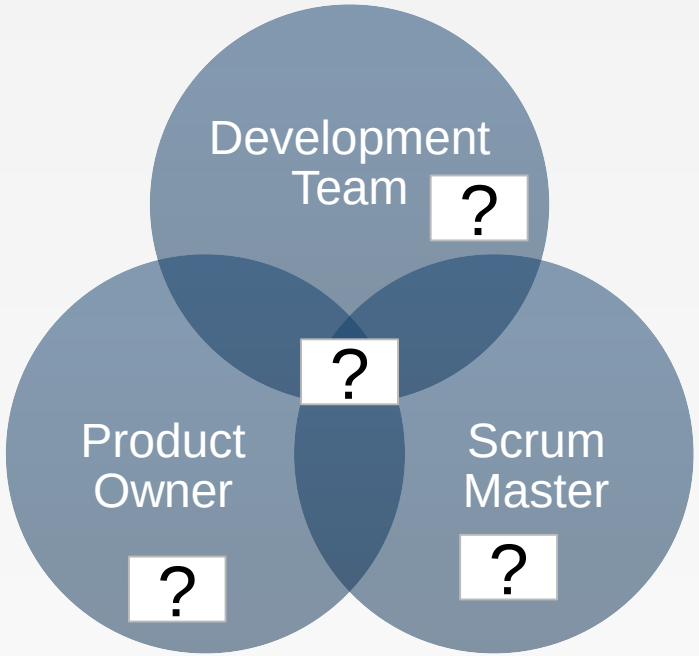
- Member of the Agile Team
- A single voice for the Customer and stakeholders
- Owns and manages the Team Backlog
- Defines and accepts requirements
- Makes the hard calls on scope and content

The Product Owner in the Enterprise

- Establishes the sequence of backlog items based on program priorities, events, and dependencies with other teams
- Operates as part of an extended Product Management Team, usually reporting via a “fat dotted line” to Product Management
- Understands how the Enterprise Backlog Model operates with Epics, Capabilities, Features, and Stories
- Uses PI Objectives and Iteration Goals to communicate with management
- Coordinates with other Product Owners, the System Team, and shared services in the PI Planning meetings
- Works with other Product Owners and the Product Management team throughout each Iteration and PI

Scrum roles and responsibilities card game

- In your group, draw the following Venn diagram
- Review the responsibilities sheet in the Appendix and create sticky notes for each responsibility, placing them either in a role or at an intersection
- Prepare to discuss your decisions



PREPARE 15 min | SHARE 5 min

2.3 Meet the teams and people on the train

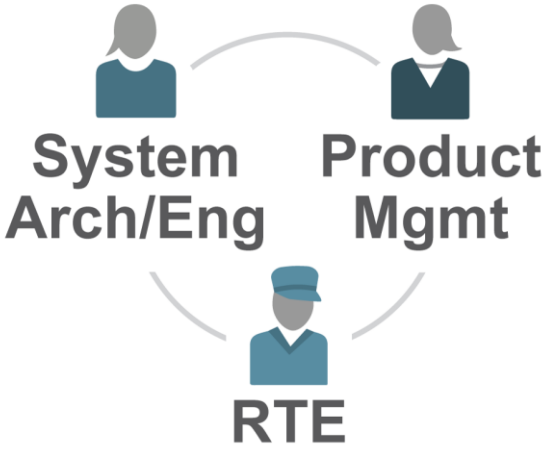
The Agile Release Train

- A virtual organization of 5 – 12 teams (50 – 125+ individuals) that plans, commits, and executes together
- Program Increment (PI) is a fixed time box; default is 10 weeks
- Synchronized Iterations and PIs
- Aligned to a common mission via a single Program Backlog
- Operates under architectural and UX guidance
- Frequently produces valuable and evaluable System Level Solutions

The ART roles

Three primary roles help ensure successful execution of the Vision and Roadmap initi
Level:

1. The Release Train Engineer is a servant leader who facilitates and guides the work of the ART. He acts like a chief Scrum Master.
2. Product Management is the main content authority guiding the train. They own and prioritize the Program Backlog.
3. The System Architect/Engineer has the technical responsibility for the overall architectural and engineering design of the system. She provides architectural and technical guidance to the teams on the train.



Exercise: Know the people on the train

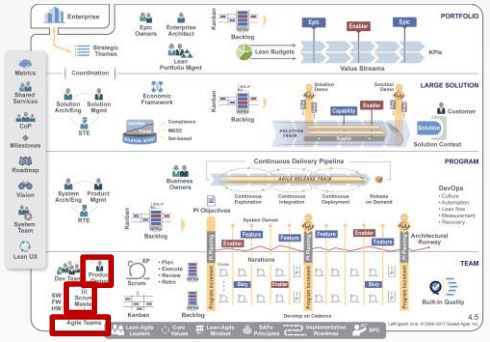
The RTE presents himself and the main players on the train:

- Product Management
- System Architect/Engineering
- Lean UX
- DevOps
- Shared Services
- Each team presents itself (name, area of responsibility, special skills)

Lesson summary

In this lesson, you:

- Built your team and learned about their roles
- Explored the roles of the Scrum Master and the Product Owner
- Met the people and teams on the train and learned about their roles



Suggested Scaled Agile Framework reading:

- *"Agile Teams" article*
- *"Scrum Master" article*
- *"Product Owner" article*

Lesson 3
Planning the Iteration

1. Introducing the Scaled Agile Framework
2. Building an Agile Team
3. Planning the Iteration
4. Executing the Iteration
5. Executing the PI

SAFe® Course Attending this course gives students access to the SAFe Practitioner exam and related preparation materials.

Learning objectives

3.1 Prepare the Backlog

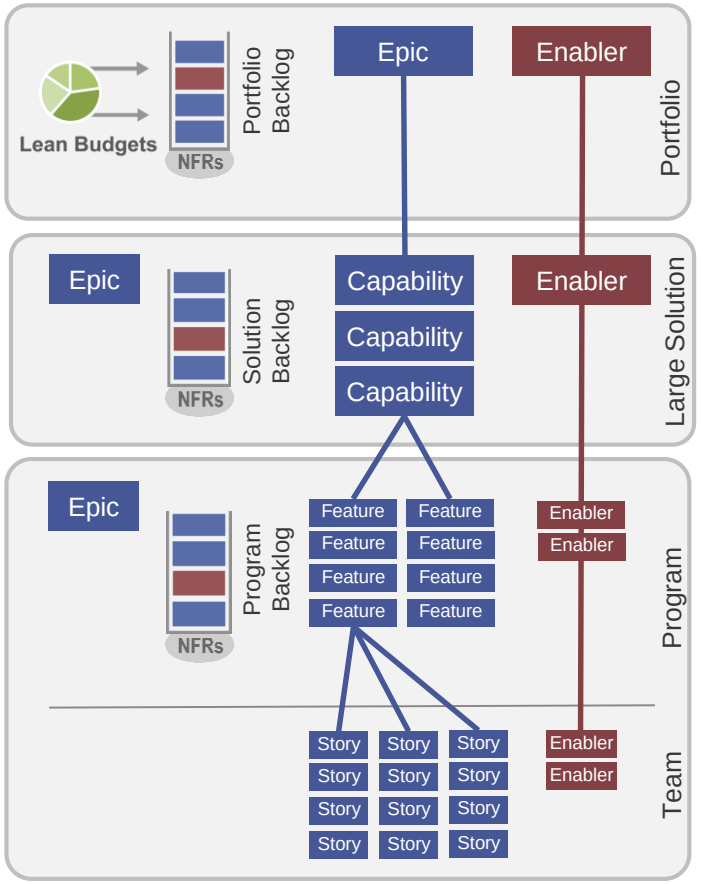
3.2 Plan the Iteration

3.1 Prepare the Backlog

Define Solution Features for the Program Backlog

Features are services that fulfill user needs.

- “Feature” is an industry-standard term familiar to marketing and Product Management
- Expressed as a phrase, “value” is expressed in terms of benefits
- Identified, prioritized, estimated, and maintained in the Program Backlog



Features have benefits and acceptance criteria

- Benefit hypothesis justifies Feature implementation cost and provides business perspective when making scope decisions
- Business benefits impact economic prioritization of the Feature
- Acceptance criteria is typically defined during Program Backlog refinement
- Reflect functional and nonfunctional requirements

SSO example:

Multifactor authentication

Benefit hypothesis

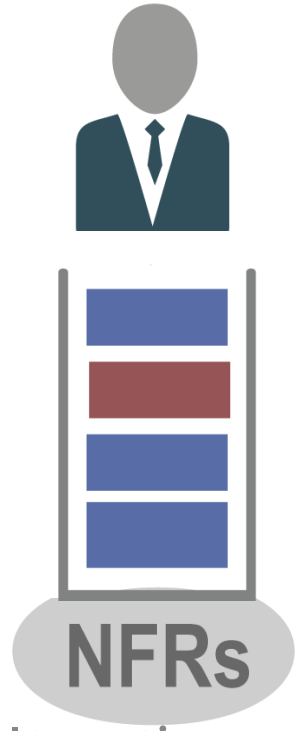
Enhance user security via both password and a device.

Acceptance criteria

1. USB tokens as a first layer
2. Password authentication as a second layer
3. Multiple tokens on a single device
4. User activity log reflecting both authentication factors

The Team Backlog

- Contains all the work the team needs to work on
- Created by the Product Owner and the team
- Prioritized by the Product Owner
- Contains User and Enabler Stories
  - User stories provide Customers with value
  - Enabler Stories build the infrastructure and architectures that makes user stories possible
- Stories in the backlog are prioritized
- Stories for the next Iteration are more detailed than Stories for later Iterations
- Nonfunctional requirements are a constraint on the backlog



User stories

- Containers for user or Customer value
- Written using the following template:

As a <user role> I want <activity> so that <business value>

- **User role** is the description of the person doing the action
- **Activity** is what they can do with the system
- **Business value** is why they want to do the activity

As a driver, I want to limit the amount of money before I fuel so that I can control my expenditure

As a driver, I want a receipt after fueling so that I can expense the purchase

As the Finance Department, we want print receipts only for drivers who request them so that we can save on paper

(Roles can be people, devices, or systems)

User Story guidelines — The 3 Cs

Card

They are written on a card or in the tool and may annotate with notes.

As a spouse,
I want a clean garage so that I can park my car and not trip on my way to the door.

Conversation

The details are in a conversation with the Product Owner.

What about the bikes?

Oh yeah, hang the bikes

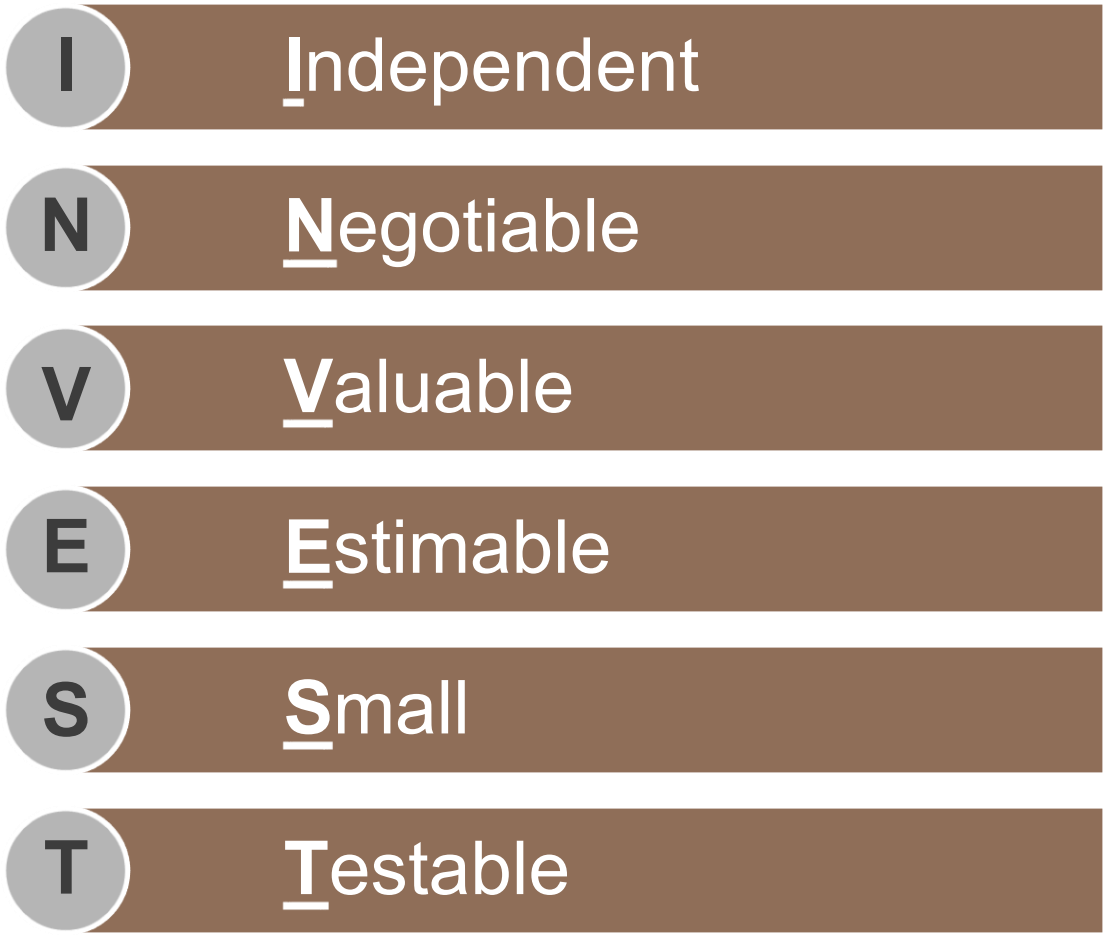
Confirmation

Acceptance criteria confirm the story correctness.

- Tools have been put away
- Items on the floor have been returned to the proper shelf
- Bikes have been hung

Source: 3 Cs coined by Ron Jeffries

INVEST in a good Story



Enabler Stories

Enabler Stories build the groundwork for future user stories.

- Three types of Enabler Stories:
  1. Infrastructure – Build development and testing frameworks that enable a faster and more efficient development process
  2. Architecture – Build the Architectural Runway, which enables smoother and faster development
  3. Exploration – Build understanding of what is needed by the customer, to understand prospective solutions and evaluate alternatives

Splitting Features and Stories

Techniques for splitting Features and Stories to fit within their boundaries (PI and Iteration, respectively).

1. Workflow steps
2. Business rule variations
3. Major effort
4. Simple/complex
5. Variations in data
6. Data methods
7. Defer system qualities
8. Operations
9. Use-case scenarios
10. Break out a spike

Exercise: Break Features into Stories

Work with your team to break Features from the Program Backlog into Stories that enough to fit into an Iteration.

- Break a Feature into 5 to 10 Stories
- Make sure the Stories retain a business value
- Try to create Stories that are less than a week in size
- Identify spikes as needed
- You should have at least 5 Stories from your Features

Acceptance criteria

- Acceptance criteria provide the details of the Story from a testing point of view
- Acceptance criteria are created by the Team and the PO

As a driver, I want to limit the amount of money before I fuel so that I can control my expenditure.

Acceptance criteria:

1. The fueling process stops automatically on the exact value
2. I can stop fueling before the limit has been reached and will only be charged for the amount fueled

As a driver, I want a receipt after fueling so that I can expense the purchase.

Acceptance criteria :

1. Receipt includes: Amount fueled, Amount Paid, Tax, Vehicle number, Date, Time

Exercise: Write acceptance criteria

- Write acceptance criteria for the Stories you have identified
- Make sure the acceptance criteria is testable
- The Product Owner needs to approve the acceptance criteria

Estimate Stories with relative Story points

- A Story point is a singular number that represents:
  - Volume: How much is there?
  - Complexity: How hard is it?
  - Knowledge: What do we know?
  - Uncertainty: What's not known?

- Story points are relative; they are not connected to any specific unit of measure
- Compare with other Stories (an 8-point Story should take four times longer than a 2-point Story)

How *big* is it?

?

Apply Estimating Poker for fast, relative estimating

- Estimating Poker combines expert opinion, analogy, and disaggregation for quick but reliable estimates
- All team members participate

Steps

1. Each estimator gets a deck of cards
2. Reads a job
3. Estimators privately select cards
4. Cards are turned over
5. Discuss differences
6. Reestimate

Mike Cohn, *Agile Estimating and Planning*, 2005

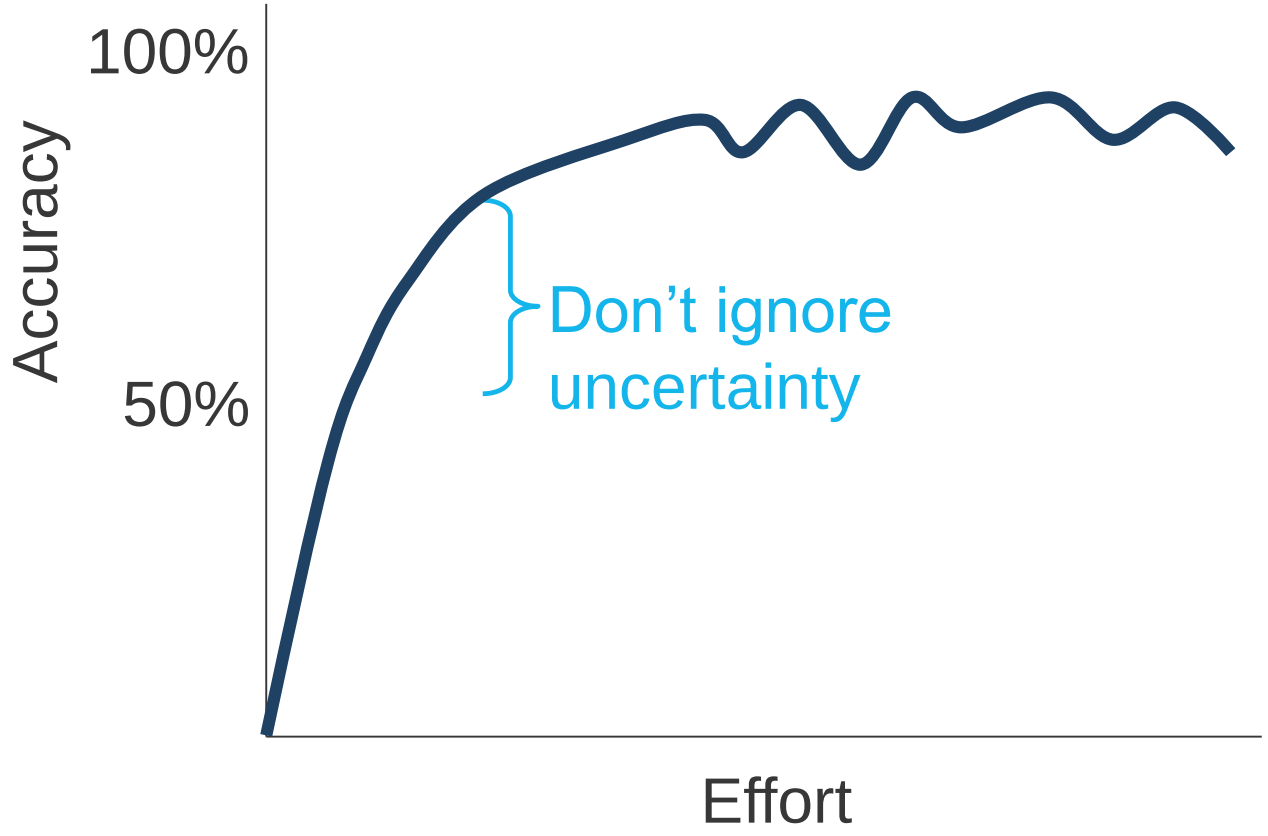
Estimation is a whole-team exercise

- Increases accuracy by including all perspectives
- Builds understanding
- Creates shared commitment

Estimation performed by a manager, architect, or select group negates these benefits.

How much time to spend estimating

A little effort helps a lot. A lot of effort only helps a little.



Exercise: Estimate Stories

- Estimate the Stories you have identified
- Use Estimating Poker together as a group
  - The Scrum Master will facilitate the activity
- The Product Owner doesn't vote but can ask or answer questions
- You should have at least three Stories estimated in Story points

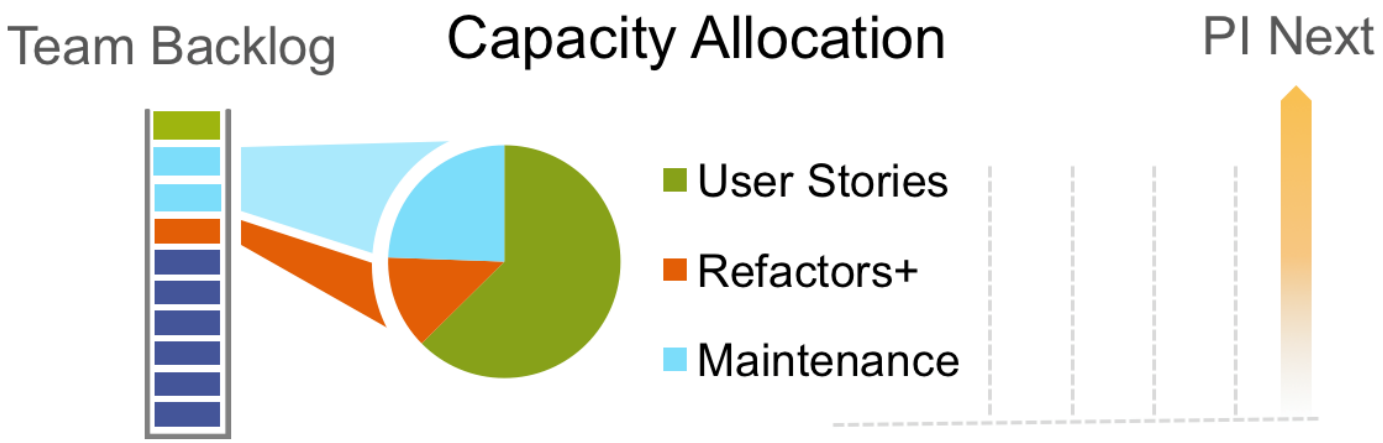
Sequencing Stories

Primary economic prioritization happens at the Program Backlog. Agile Teams sequence work for efficient execution of business priorities.

- The Product Owner and the Team sequence work based on:
  - Story priorities inherited from Program Backlog priorities
  - Events, Milestones, releases, and other commitments made during PI Planning
  - Dependencies with other teams
  - Local priorities
  - Capacity allocations for defects, maintenance, and refactors
- Initial sequencing happens during PI Planning
- Adjustments happen at Iteration boundaries

Capacity allocation for a healthy balance

- By having capacity allocation defined, the Product Owner doesn't need to prioritize unlike things against each other
- Once the capacity allocation is set, the PO and team can prioritize like things against each other



Notes:

1. Helps alleviate velocity degradation due to technical debt
2. Keeps existing Customers happy with bug fixes and enhancements
3. Can change at Iteration or PI boundaries

3.2 Plan the Iteration

Plan and commit

Purpose	Define and commit to what will be built in the Iteration
Process	▸ The Product Owner defines *what* ▸ The team defines *how* and *how much* ▸ Four hours max
Result	Iteration Goals and backlog of the team's commitment
Reciprocal commitment	▸ Team commits to delivering specific value ▸ Business commits to leaving priorities unchanged during the Iteration

Iteration Planning flow

1. The team establishes its velocity

2. The team clarifies and estimates the Stories

3. The team optionally breaks Stories into tasks

4. The process continues while there is more capacity

5. The team synthesizes Iteration Goals

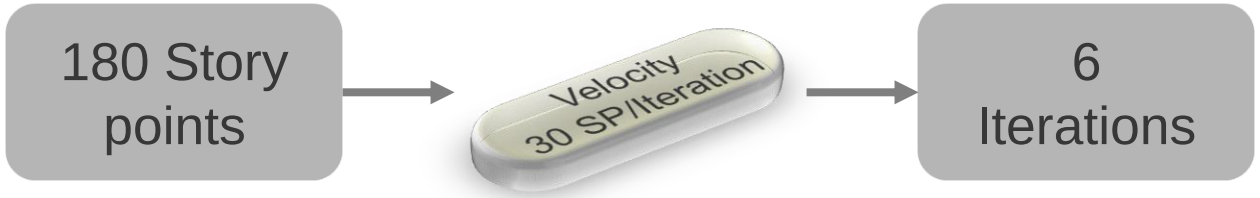
6. Everyone commits

- Timebox: 4 hours
- This meeting is by and for the team
- SMEs may attend as required

Using size to estimate duration

Establish velocity by looking at the average output of the last Iterations.

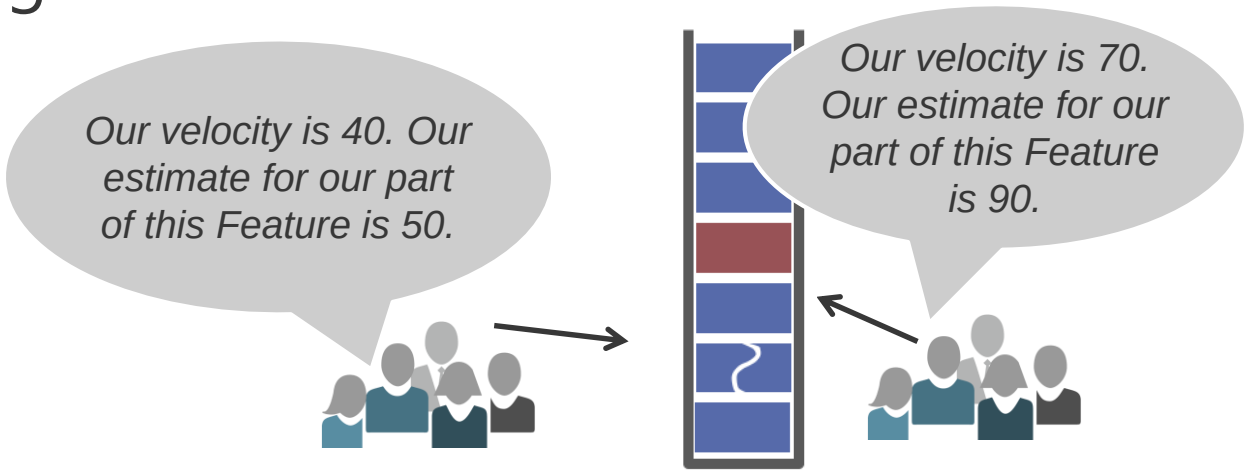
Examples



Establishing velocity before historical data exists

Normalized Story point estimating provides the economic basis for estimating work within and across programs.

1. For every full-time developer and tester on the team, give the team eight points (adjust for part-timers)
2. Subtract one point for every team member vacation day and holiday
3. Find a small Story that would take about a half-day to code and a half-day to test and validate. Call it a **1**.
4. Estimate every other Story relative to that one
5. Never look back (don't worry about recalibrating)



The cost of a Story point is about the same for each of these teams.

Example: Assuming a 6-person team composed of 3 developers, 2 testers, 1 PO, with no vacations, etc.

Estimated velocity = 5 * 8 pts = 40 pts/Iteration

Exercise: Calculate your initial velocity

Use the slide on the previous page to calculate your team's starting velocity.

- Calculate your estimated velocity for the next two-week Iteration, which starts after PI Planning
- Use your real availability
- Each team should have their estimated velocity for two Iterations

Iteration Goals

Iteration Goals provide clarity, commitment, and management information.

They serve three purposes:

1. Align team members to a common purpose
2. Align Program Teams to common PI Objectives and manage dependencies
3. Provide continuous management information

Iteration Goals example

1. Finalize and push last-name search and first-name morphology
2. Index 80% of remaining data
3. Other Stories:
   - Establish search replication validation protocol
   - Refactor artifact dictionary schema

Story analysis and estimation

- The Product Owner presents Stories in order of priority
- Each Story
  - Is discussed and analyzed by the team
  - Has its acceptance criteria refined
  - Is estimated
- The process continues until the estimation of the Stories reached the velocity of the team

Exercise: Plan your first Iteration

Use the team's initial velocity and the estimated Stories to allocate Stories to the first Iteration.

- Plan which Stories fit into the first Iteration
- Estimate more Stories as needed to fill the first Iteration
- Each team should have a list of Stories that they can deliver by the end of the first Iteration

Commit to the Iteration Goals

A Team meets its commitment:

By doing everything they said they would do.

- or -

In the event that it is not feasible, they must immediately raise a red flag.

Commitment

Too much holding to a commitment can lead to burnout, inflexibility, and quality problems.

Adaptability

Too little commitment can lead to unpredictability and lack of focus on results.

Team commitments are not just to the work. They are committed to other teams, the program, and the stakeholders.

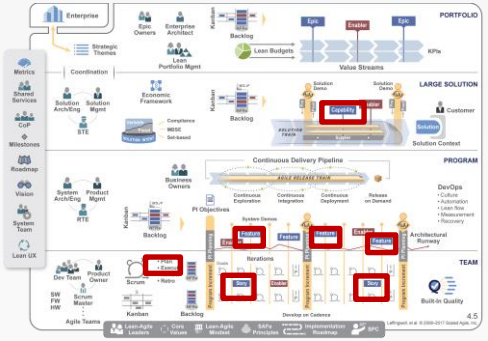
Iteration planning for Kanban teams

- Some teams have a more responsive nature to their work, such as maintenance teams, System Teams, DevOps
- These teams find less value in trying to plan the Iteration in detail
- Kanban teams still publish Iteration Goals, which consist of the known parts of their work
- They commit to the goals as well as to a cycle time SLA for incoming work based on their known historical data

Lesson summary

In this lesson, you:

- Prepared your backlog of stories through breaking down features
- Planned your iteration using story estimation



Suggested Scaled Agile Framework reading:

- *"Stories" article*
- *"Features and Capabilities" article*
- *"Iteration Planning" article*

Lesson 4
Executing the Iteration

1. Introducing the Scaled Agile Framework
2. Building an Agile Team
3. Planning the Iteration
4. Executing the Iteration
5. Executing the PI

SAFe® Course Attending this course gives students access to the SAFe Practitioner exam and related preparation materials.

Learning objectives

4.1 Visualize the flow of work

4.2 Measure the flow of work

4.3 Build quality in

4.4 Continuously integrate, deploy, and release

4.5 Control flow with meetings

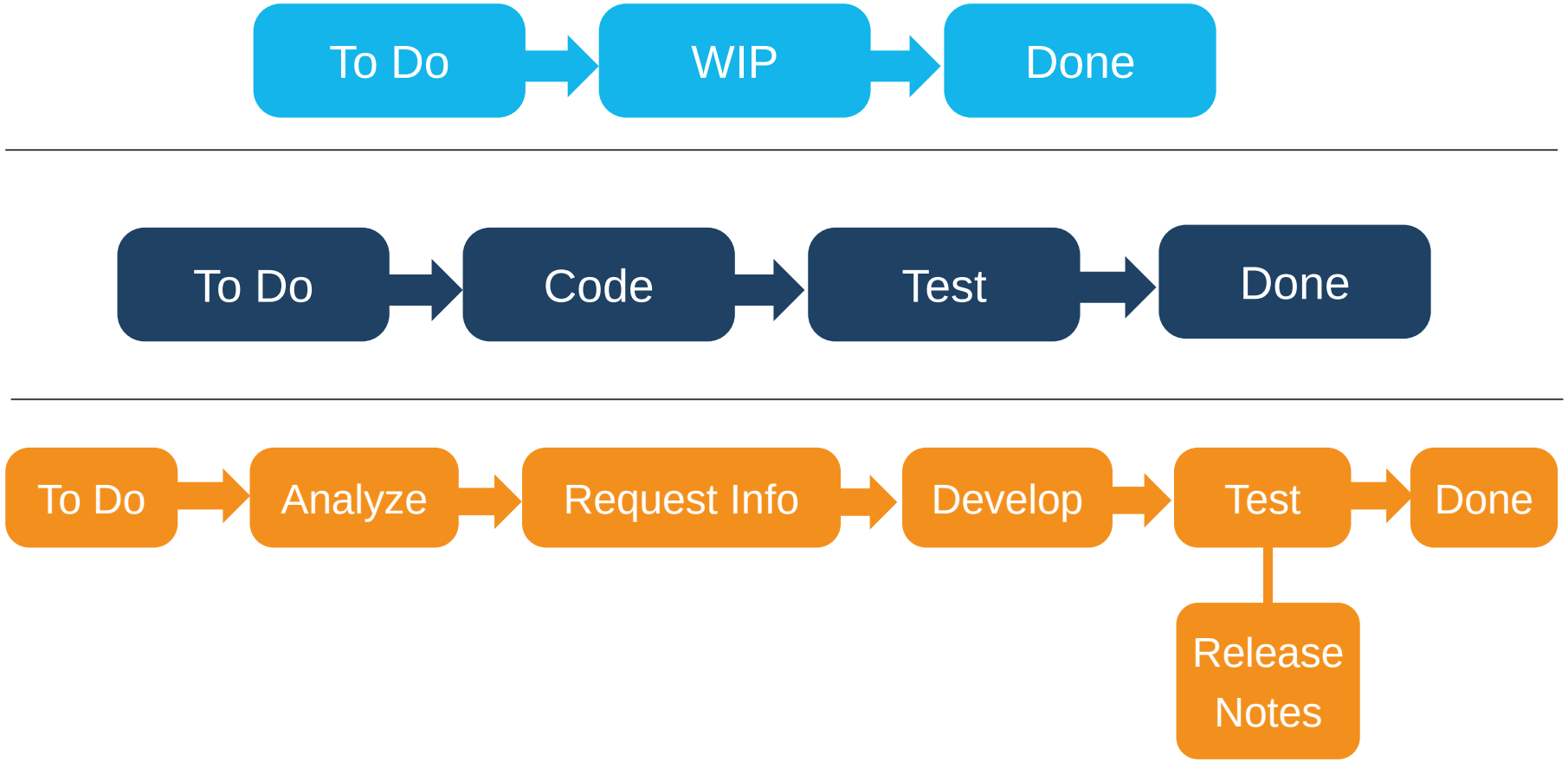
4.6 Demonstrate value

4.7 Retrospect and improve

4.1 Visualize the flow of work

Visualize the flow of work

- What is the flow of work for your team?
- What are the steps it takes to get a Story to Done?



Setting WIP limits

WIP limits can apply to a single step or to multiple steps. Some steps have no WIP limits, while others serve as buffers and have minimal as well as maximal WIP.

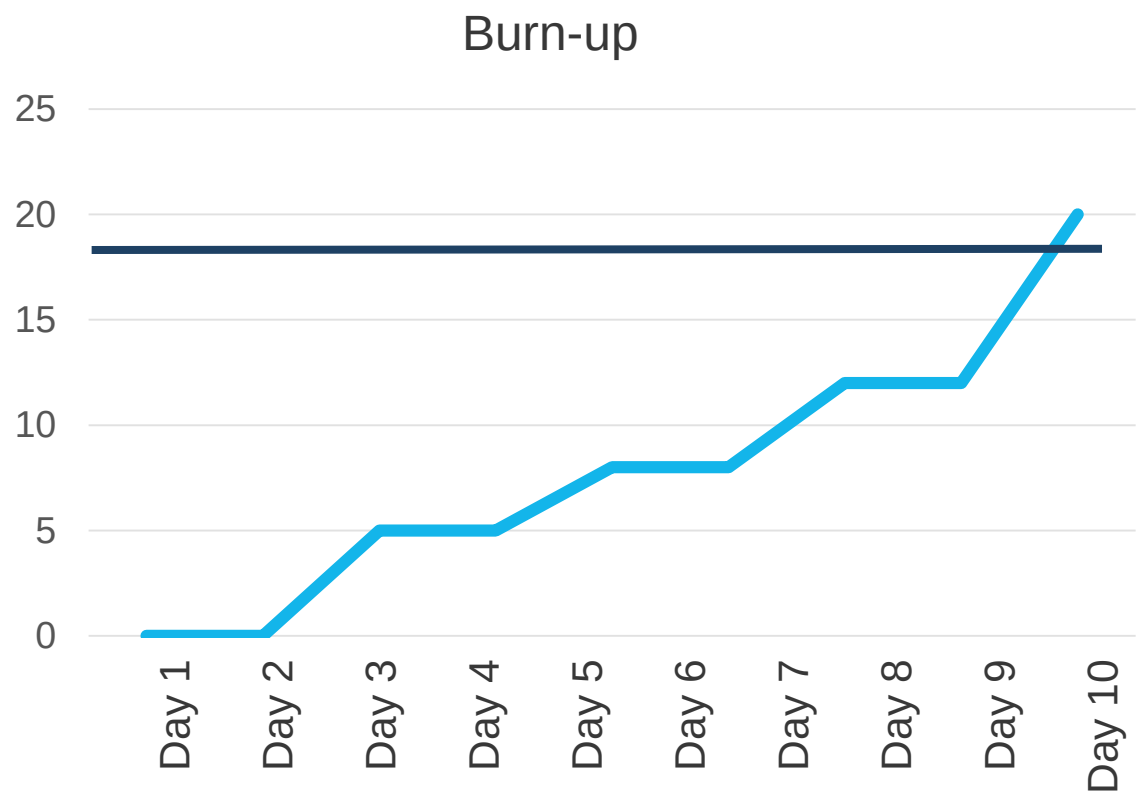
Exercise: Build your initial board

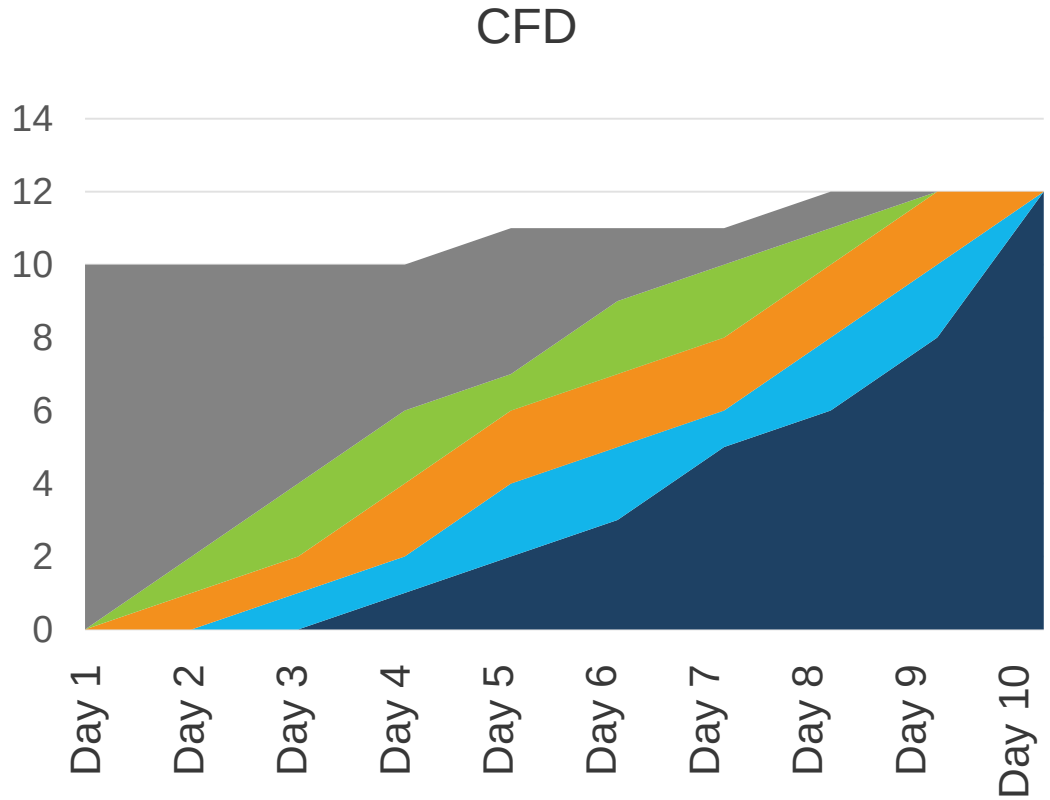
Consider the previous slides and build an initial board for your team.

- Define the steps you need to turn Stories into value
- Build the structure of the board
- Assign initial WIP limits

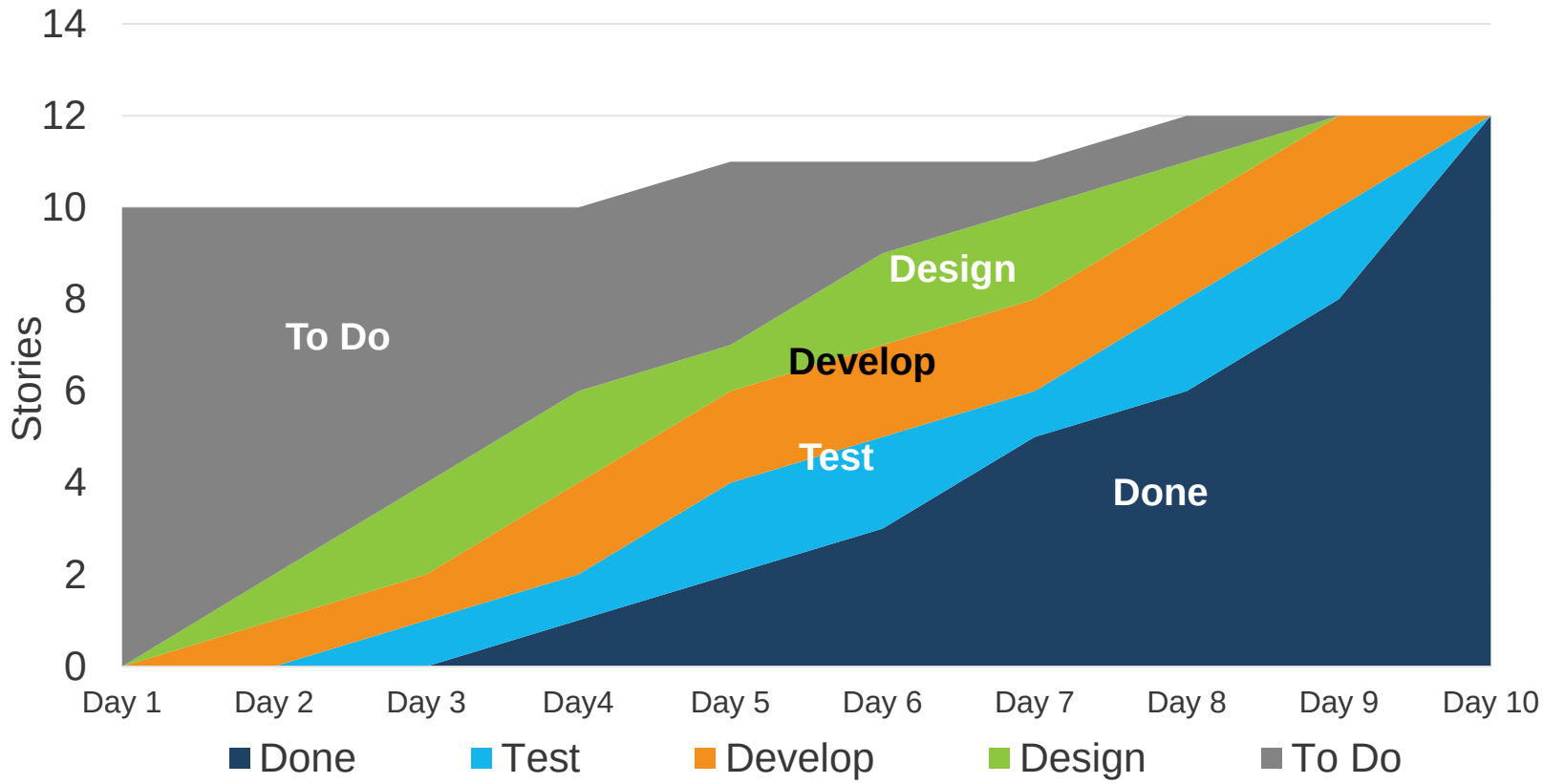
4.2 Measure the flow of value

Track status with burn-up charts and CFDs

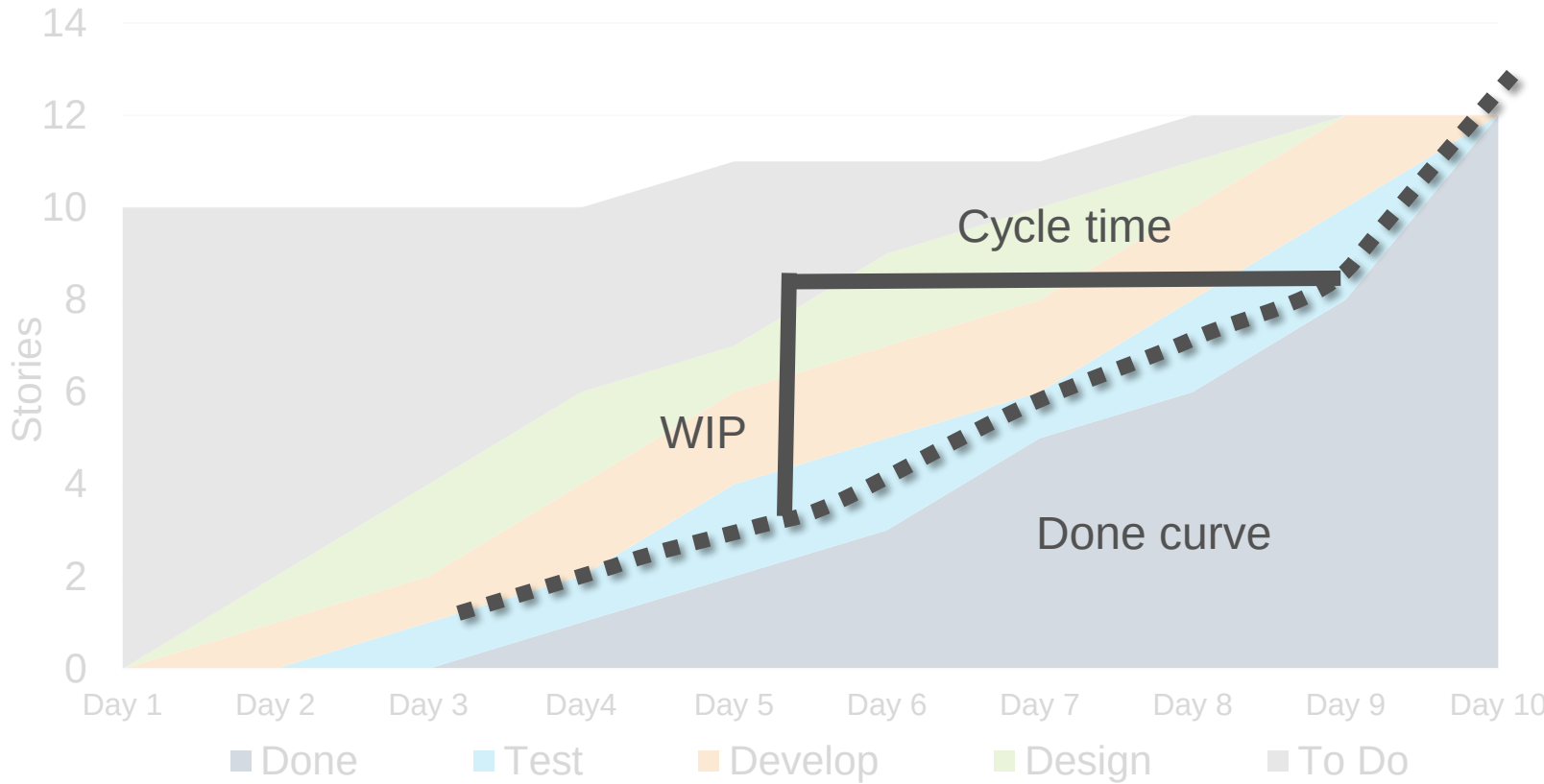




Understand Cumulative Flow Diagrams (CFD)



What can you learn from a CFD?



4.3 Build quality in

Exercise: Build quality in poster

- Review all the slides in the following section (10 slides)
- Create a poster to reflect the different aspects of building quality in
- Indicate what your team can do to incorporate them into your work

PREPARE

SHARE

Built-In Quality

"You can't scale crappy code" (or hardware, or anything else)

Building quality in:

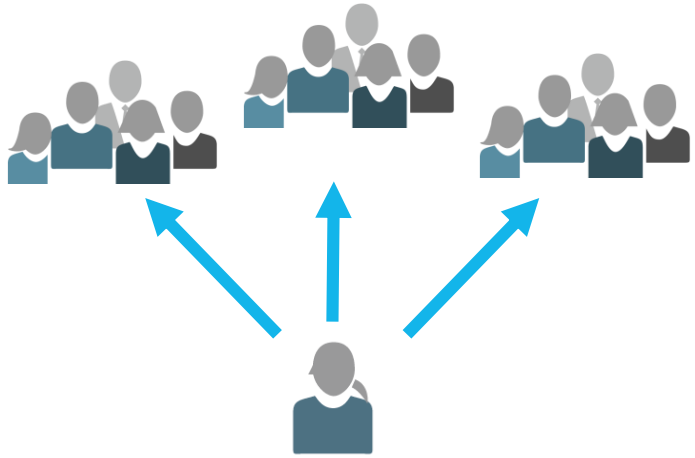
- Ensures that every increment of the Solution reflects quality standards
- Is required for high, sustainable development velocity
- Software quality practices (most inspired by XP) include Continuous Integration, Test-First, refactoring, pair work, collective ownership, and more
- Hardware quality is supported by exploratory, early Iterations; frequent system-level integration; design verification; MBSE; and Set-Based Design

Built-In Quality

Emergent design and intentional architecture

Every team deserves to see the bigger picture. Every team is empowered to design their part.

- Emergent design – Teams grow the system design as user stories require
- Intentional architecture – Fosters team alignment and defines the Architectural Runway

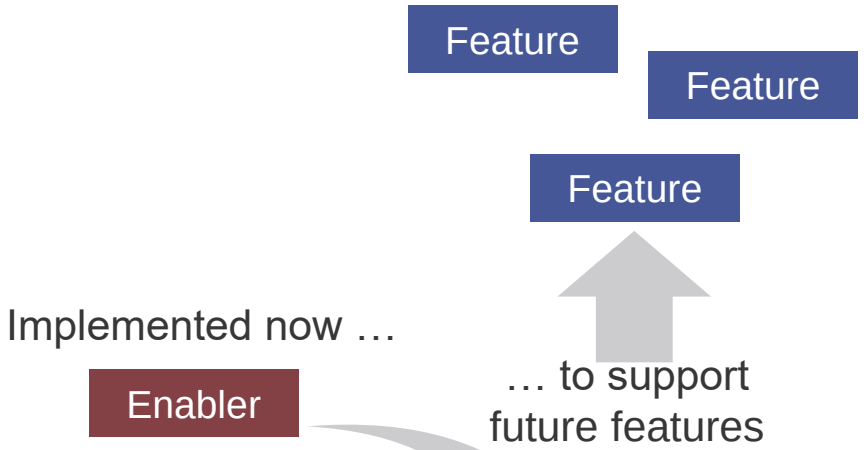


A balance between emergent design and intentional architecture is required for speed of development and maintainability.

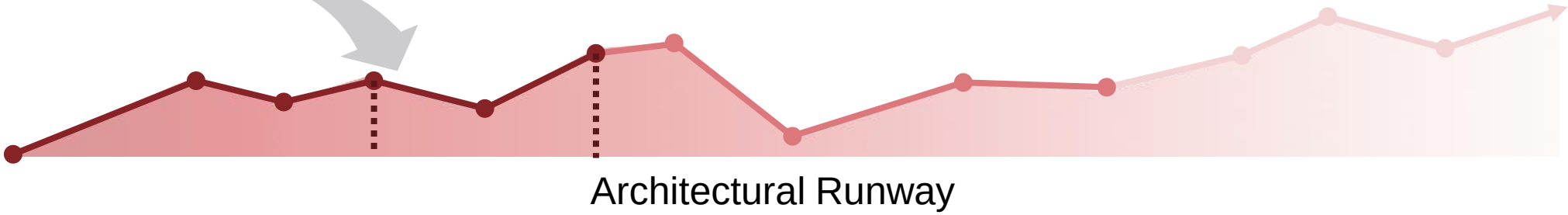
Architectural Runway

Architectural Runway is existing code, hardware components, etc. that technically enable near-term business features.

Example: A new, fuzzy search algorithm will enable a variety of future features that can accept potentially erroneous user input



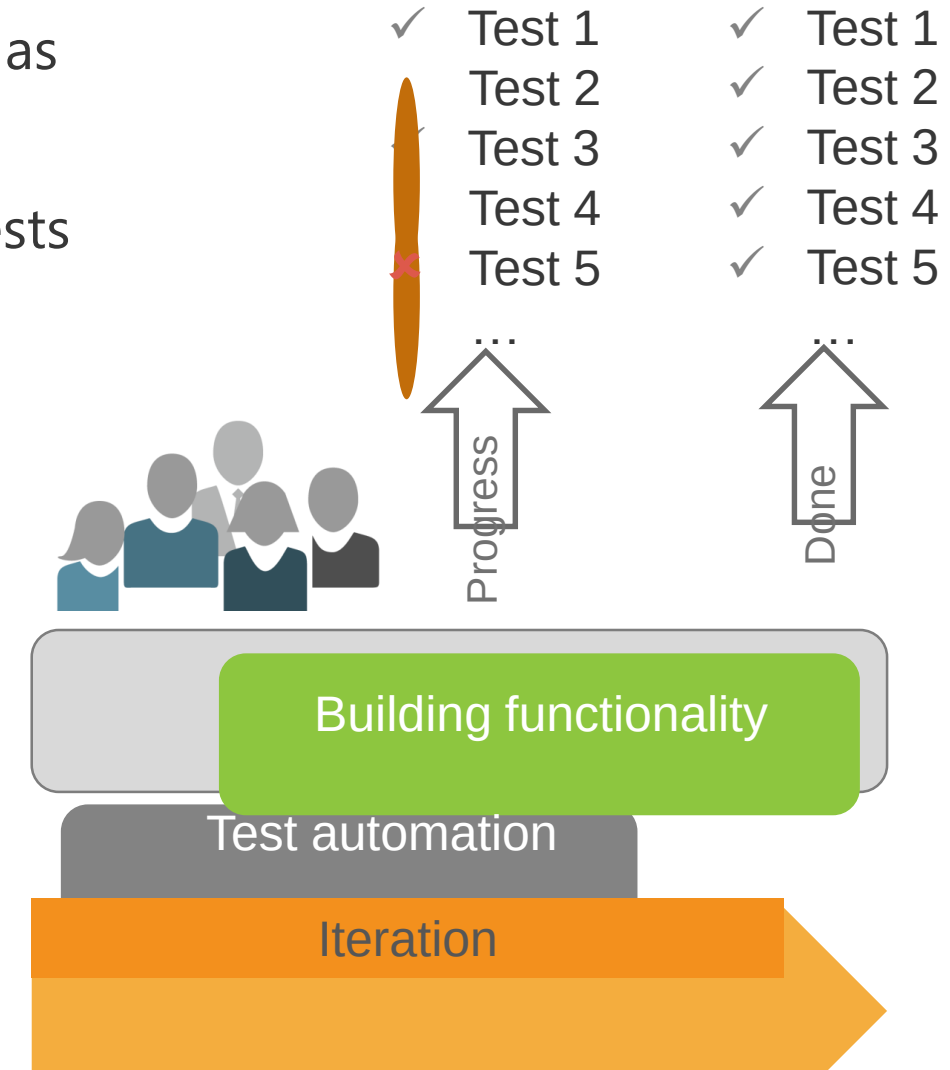
- Enablers build up the runway
- Features consume it
- Architectural Runway must be continuously maintained
- Use Capacity Allocation (a percentage of train's overall capacity in a PI) for Enablers that extend the runway



Test first: Automate now!

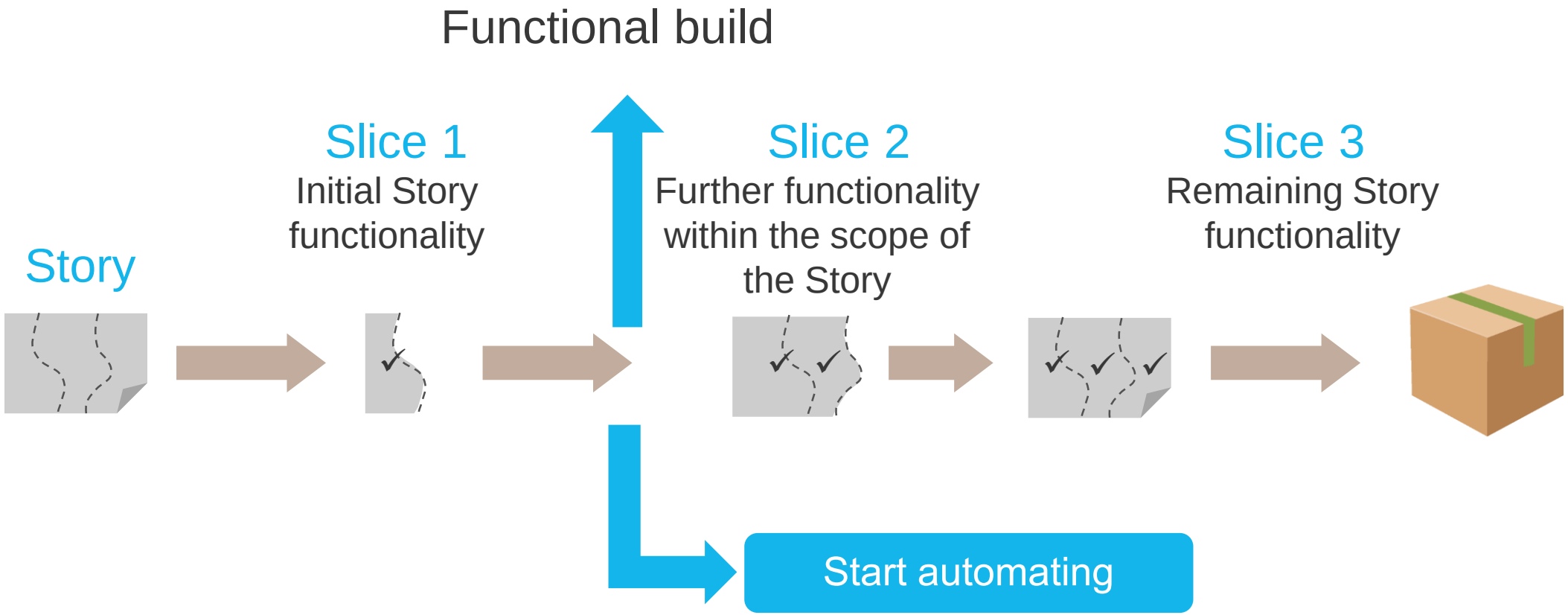
Else, velocity is bottlenecked, quality is speculative, scaling is impossible

- Automated tests are implemented in the same iteration as the functionality
- The team that builds functionality also automates the tests
- Create an isolated automated test environment
- Actively maintain test data under version control
- Passing vs. not-yet-passing and broken automated tests are the *real* iteration progress indicator



Transition to early automation

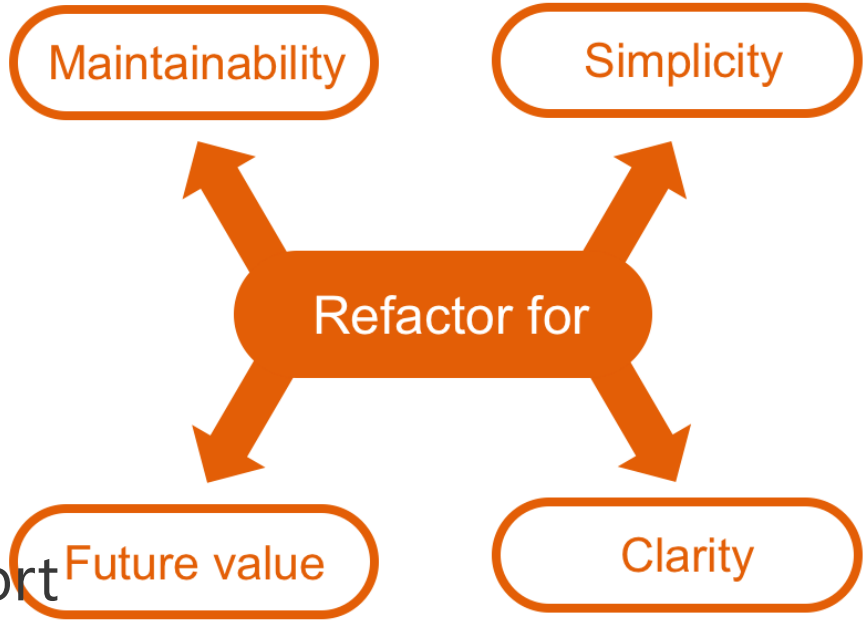
Start automating as soon as the initial functionality is in place.



Refactoring

Refactoring allows teams to maintain high velocity.

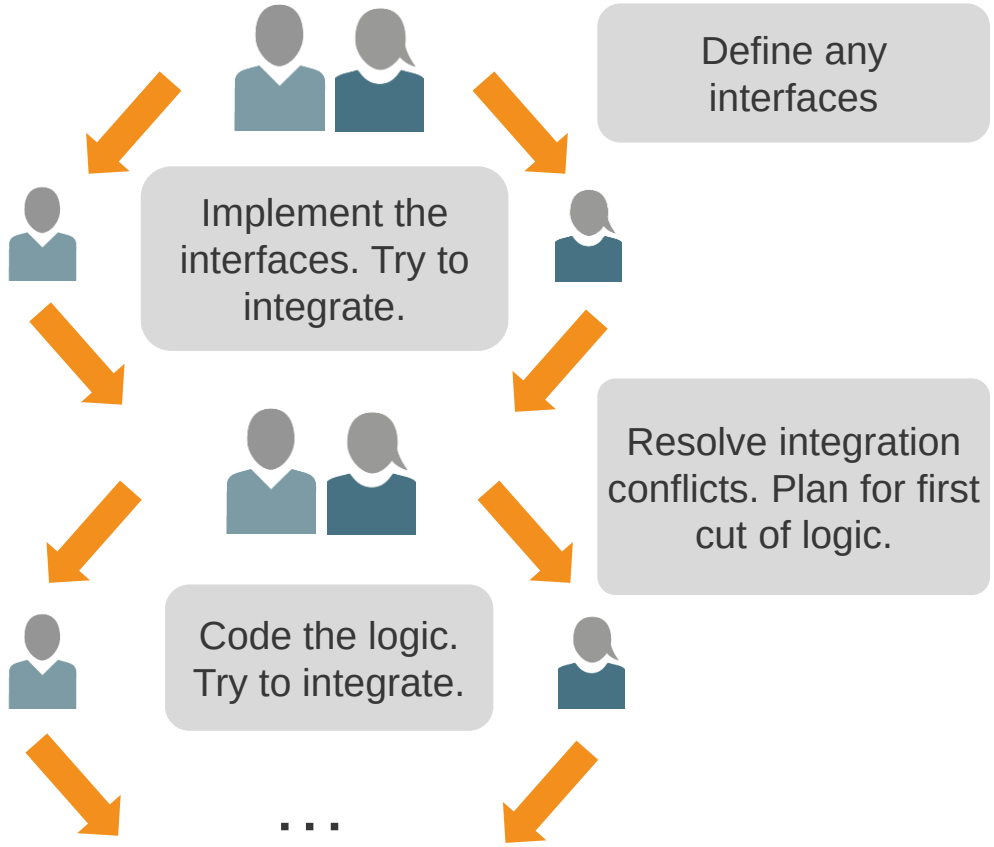
- It is impossible to predict requirements or design in detail
- Refactoring allows teams to quickly correct the course of action
- Emergent design is impossible without continuous refactoring
- Most user stories will include some refactoring effort
- If technical debt is big – teams track and implement as separate backlog items – then refactor



Pair work

Pair work improves system quality, design decisions, knowledge sharing, and team velocity.

- Pair work is …
  - Broader and less constraining than pair programming
  - A collaborative effort of any two team members: dev/dev, dev/PO, dev/tester, etc.
- Team members spend 20% to 80% time pairing
- Spontaneous pairing, and purposeful rotation over time

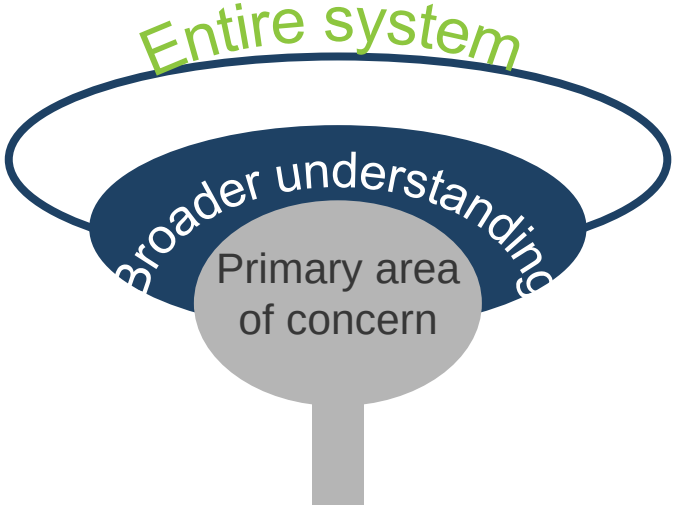


Example user story implementation flow

Collective ownership

Collective ownership addresses bottlenecks, increases velocity, and encourages shared contribution.

- Collective ownership fosters Feature orientation
- Collective test ownership is even more important—it facilitates shared understanding of system behavior
- Collective ownership is supported by:
  - Design simplicity
  - Communities of Practice
  - Pair work
  - Joint specification and design workshops
  - Frequent integration of the entire system
  - Standards



Agile Team

Teams gain a broader understanding of the system

Capture with models, not documents

- Models are more adaptable than documents
  - Single changes propagate to all uses
  - Everyone can contribute—all domains, team members
  - Faster, and automated, verification and validation

- Models are more powerful than documents
  - Increases rigor, consistency, accuracy
  - Enables earlier verification and validation
  - Facilitates more strategic reuse
  - Automates document generation (compliance)
  - Provides traceability

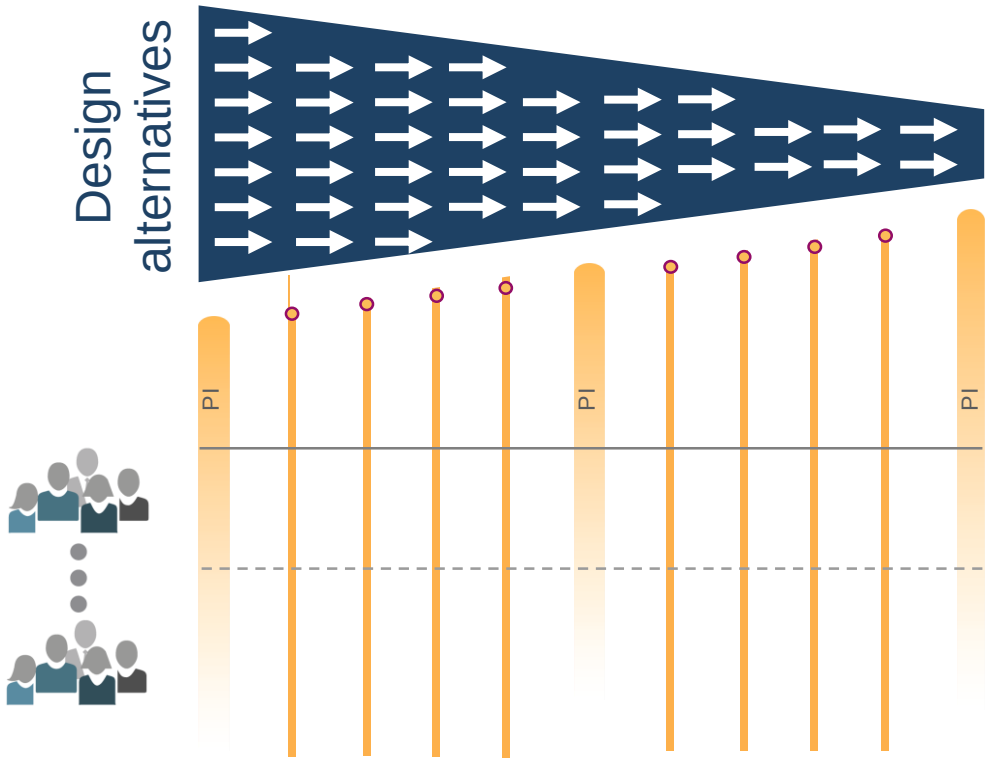
MBSE

Model-Based Systems Engineering

Set-Based Design

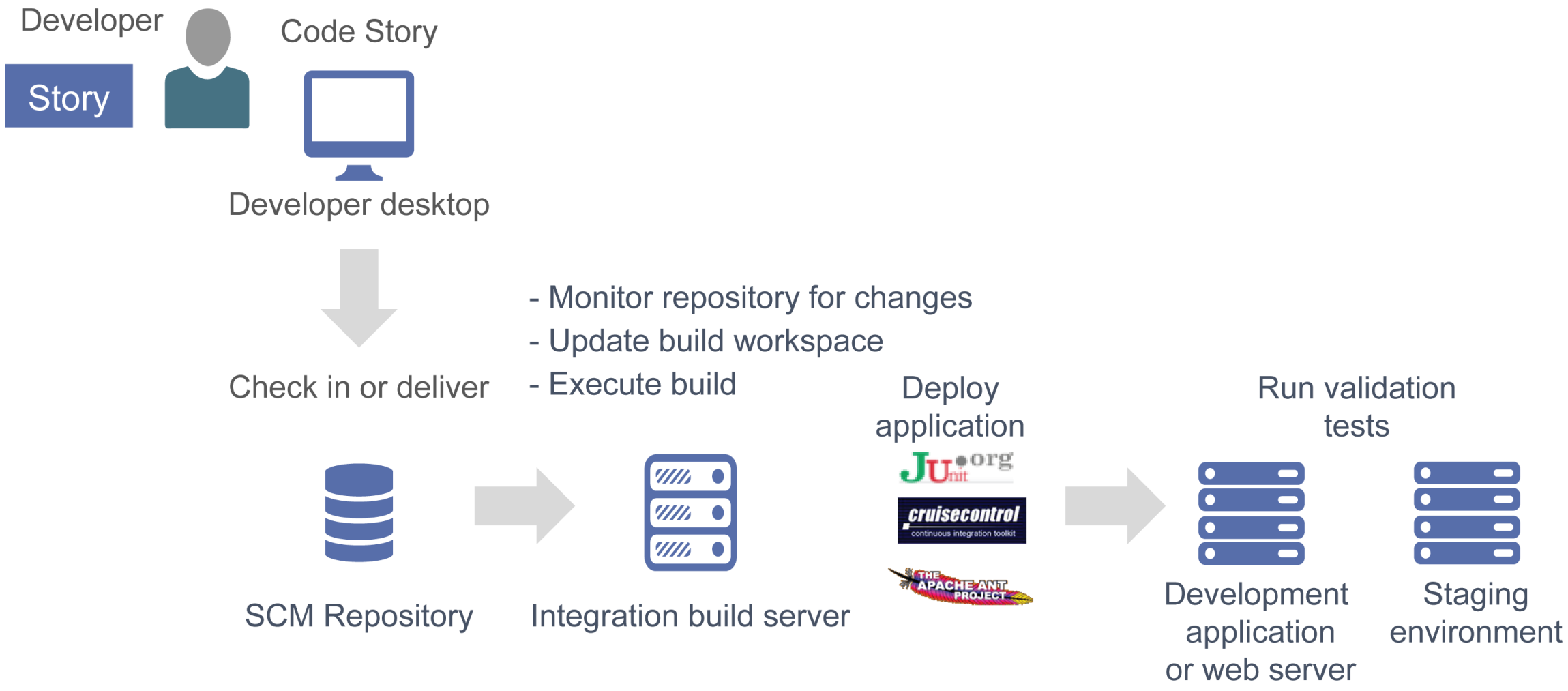
Continuously convert uncertainty to knowledge.

- Emphasizes design *discovery* over design *creation*
- Concurrently explores multiple designs options to find an optimum, rather than the first, Solution
- Understands design trade-offs before detailing specifications



4.4 Continuously integrate, deploy and release

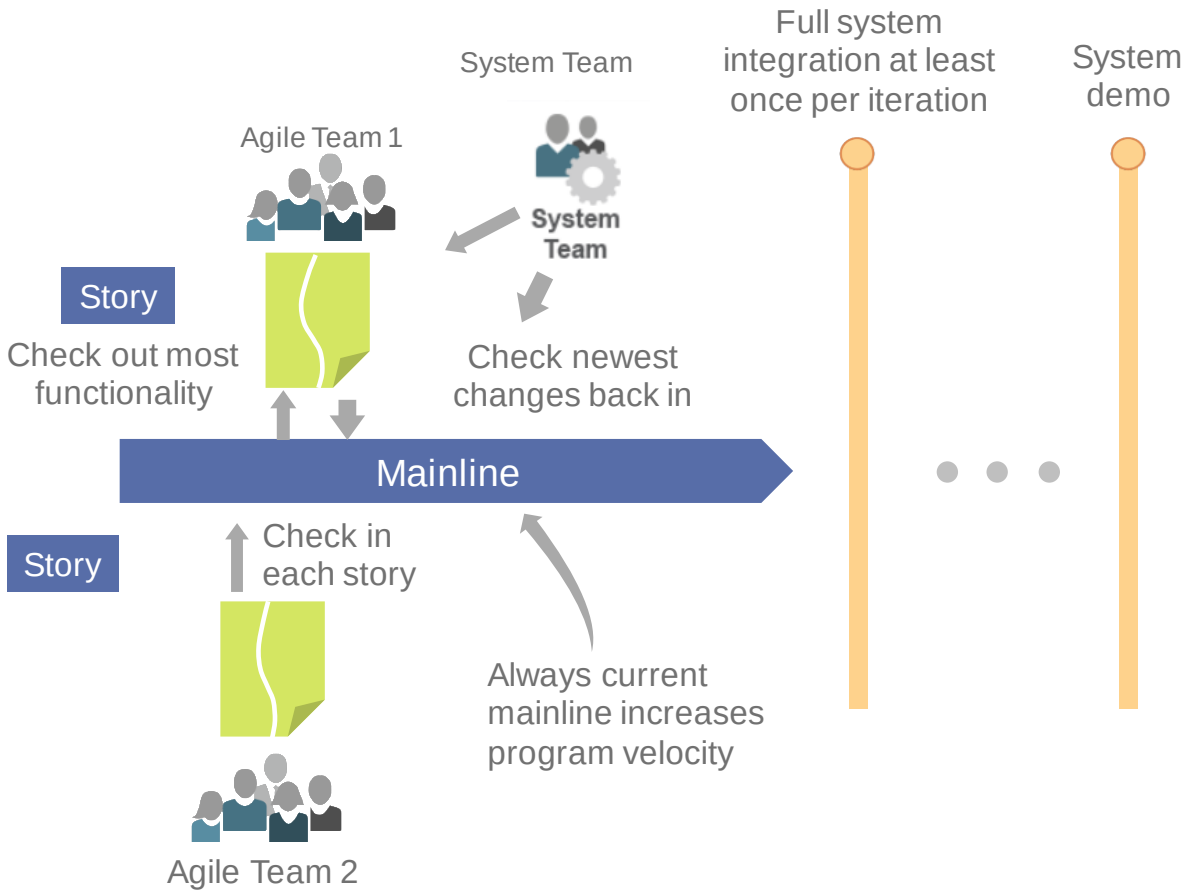
Continuous story integration



Continuous system integration

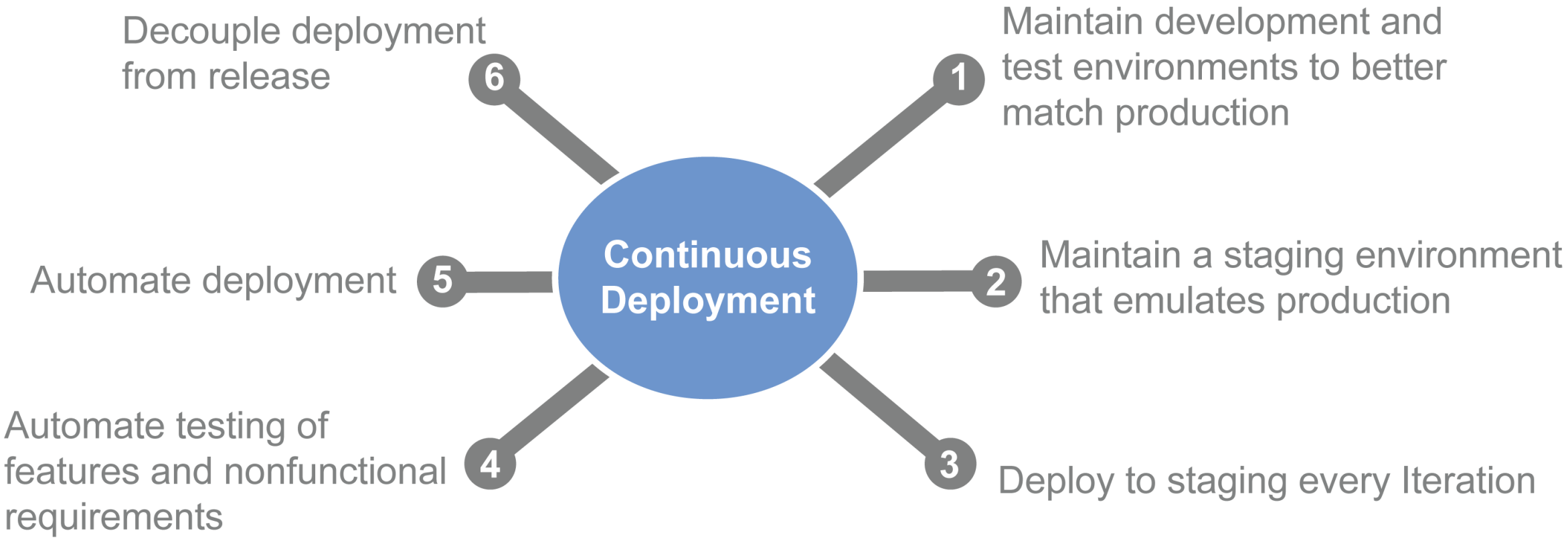
Teams continuously integrate assets (leaving as little as possible to the System Team).

- Integrate every vertical slice of a user story
- Avoid physical branching for software
- Frequently integrate hardware branches
- Use development by intention in case of inter-team dependencies
  - Define interfaces and integrate first; then add functionality



Six Recommended Practices for Continuous Deployment (CD)

CD is an important practice for every team member, team and ART.



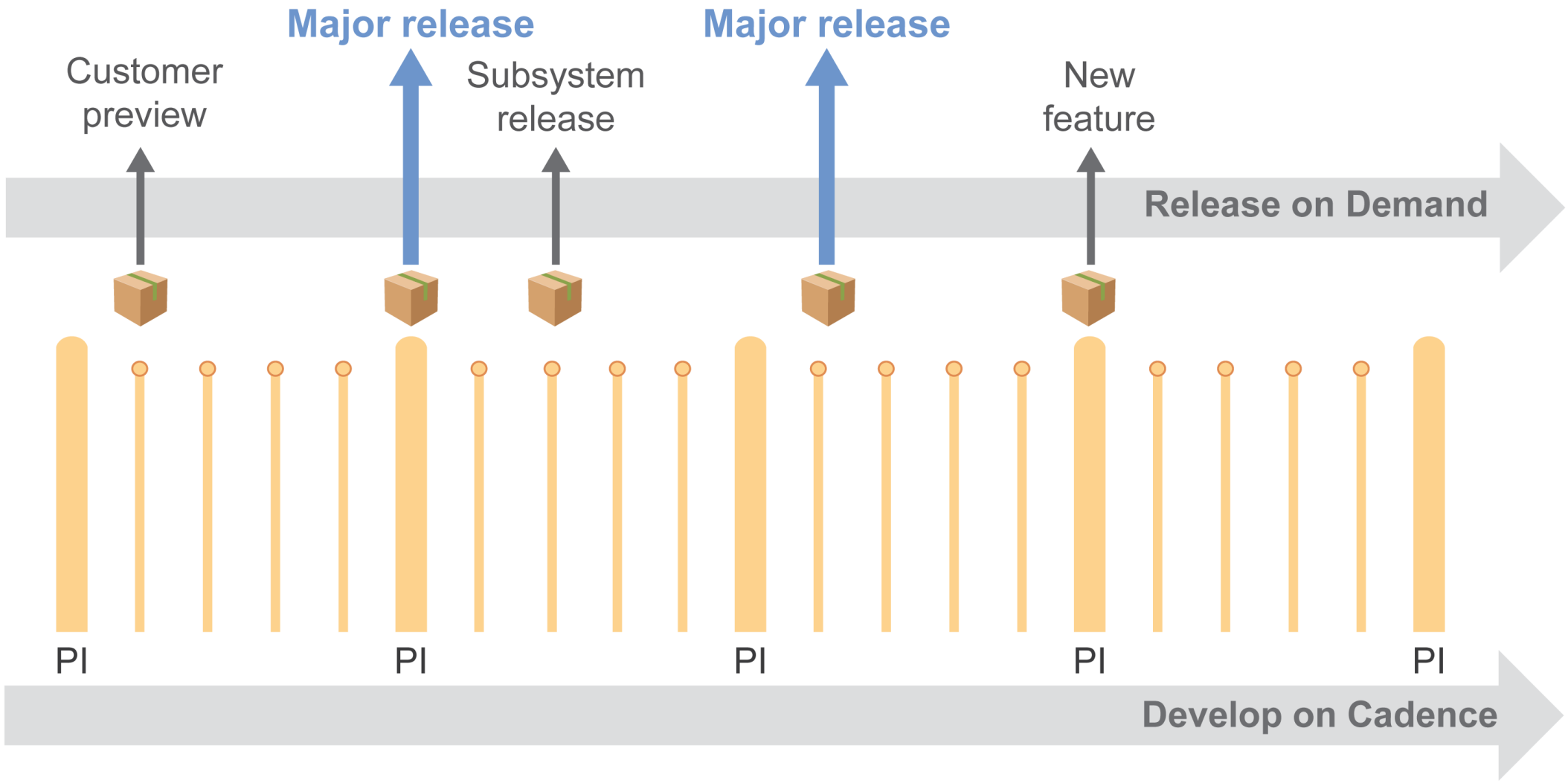
Exercise: Continuous integration and deployment

- What are the challenges to continuously integrating?
- What are the challenges to continuously deploying?
- Think about various aspects: environment, culture, tools and people
- In your group, prepare a list of 3 to 5 items that would make it hard to continuously integrate and deploy in your organization and how to solve them
- Be ready to present to the class

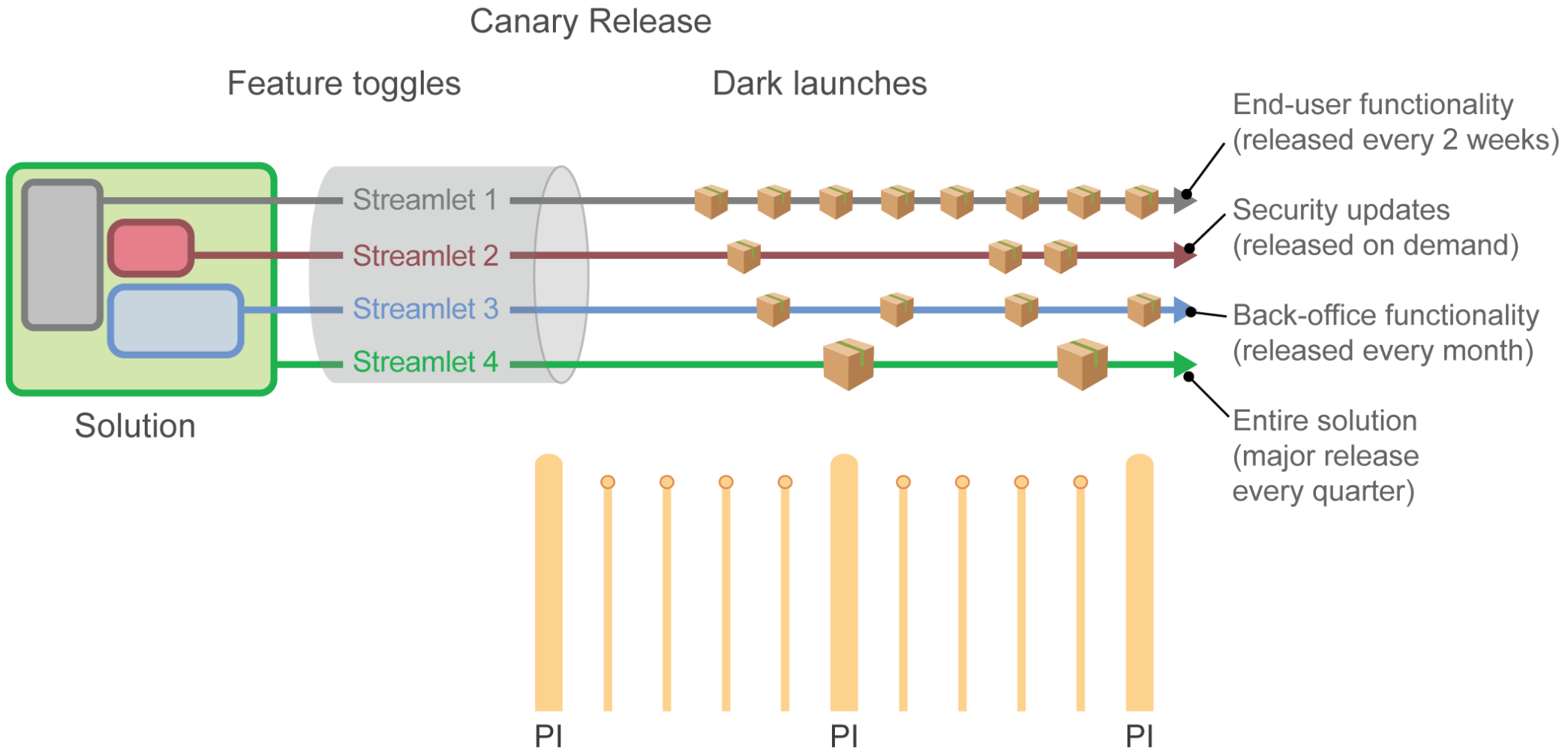
PREPARE

SHARE

Develop on Cadence. Release on Demand Time.



Decouple the release from the solution



Continuous Delivery Culture

- Watch the Bing video
- What cultural changes did the organization need to go through?
- How is your culture or environment ready for continuous delivery?
- Discuss at your table and be prepared to share

Bing: Continuous Delivery
https://youtu.be/3sFT7tgyEQk
3:28

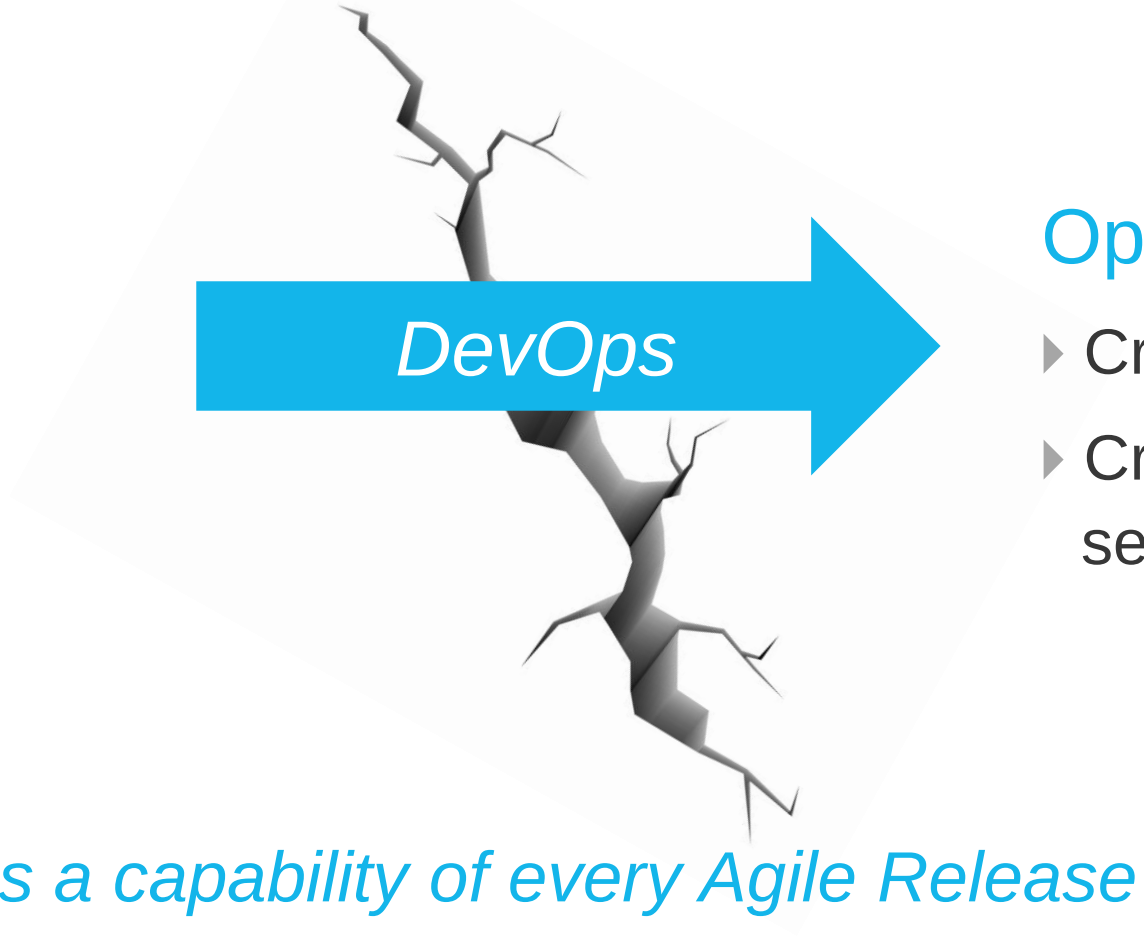
PREPARE 5 min | SHARE 3 min

What is DevOps?

DevOps is an agile approach to bridge the gap between development and operations to deliver value *faster and more reliable*.

Development:

- Create Change
- Add or Modify Features



Operations:

- Create Stability
- Create or enhance services

DevOps is a capability of every Agile Release Train

A CALMR approach to DevOps

Culture Establish a culture of shared responsibility for development, deployment and operations

Automation Automate the continuous delivery pipeline

Lean flow Keep batch sizes small, limit WIP and provide extreme visibility

Measurement Measure the flow through the pipeline. Implement application telemetry.

Recovery Architect and enable low risk releases. Establish fast recovery, fast reversion, and fast fix-forward.

4.5 Control flow with meetings

Leading the daily stand-up (DSU)

The DSU is the key to team synchronization and self-organization.

- The DSU (or daily Scrum) is not a daily status meeting for management
- It is used to:
  - Share information about progress
  - Coordinate activities
  - Raise blocking issues
- Meeting time is whenever is most convenient for the team

- Every day at the same time in front of the team board
- Timebox of 15 minutes
- Not a problem-solving session
- Update the board

Daily stand-up patterns

Basic Scrum pattern meeting agenda

Each person answers:

1. What did I do yesterday to advance the Iteration Goals?
2. What will I do today to advance the Iteration Goals?
3. Are there any impediments that will prevent the team from meeting the Iteration Goals?

The Meet-After agenda

1. Review topics the Scrum Master wrote on the meet-after board
2. Involved parties discuss, uninvolved people leave

Exercise: Simulate a daily stand-up meeting

- Let’s 4 or 5 volunteers to simulate a DSU in front of the class
- Instructor will play the Scrum Master role

Exercise: Agree on location and timing for the DSU

- As a team, we want to decide on the place and time for our DSU
- Agree on a time that works for everyone
- Take remote people into account

The backlog refinement session

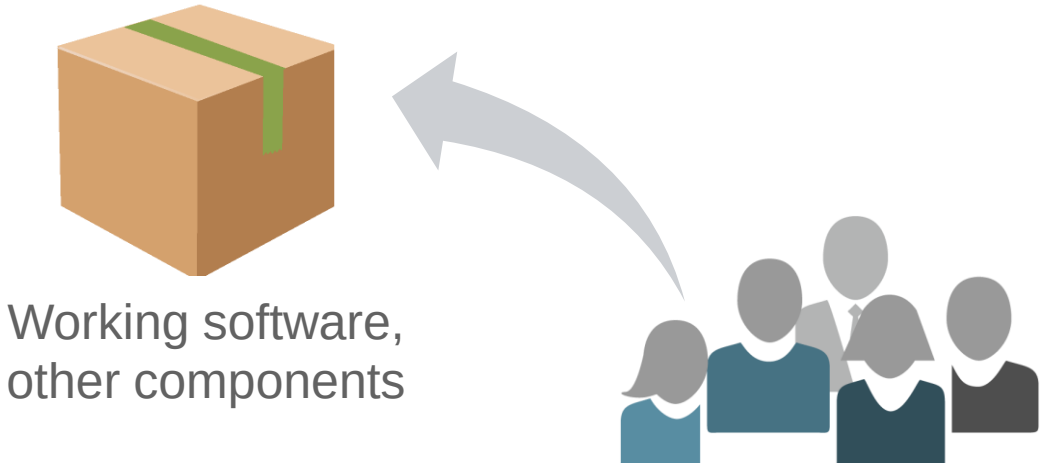
The backlog refinement session is a preview and elaboration of upcoming Stories.

- Helps the team “sleep” on new Stories prior to the Iteration Planning
- Provides enough time to identify and resolve dependencies and issues that could impact the next Iteration
- The team can improve Stories, add acceptance criteria, and point out missing information to the Product Owner
- Most of the focus is on the next Iteration, but it allows time to discuss future Iterations and even Features for the next PI

4.6 Demonstrate value

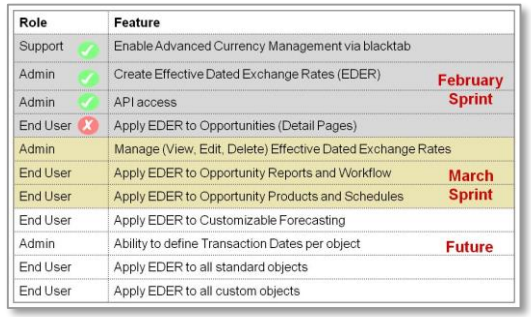
The Iteration Review

- Provides the true measure of progress by showing working software functionality, hardware components, etc.
- Preparation for the review starts with planning
- Teams demonstrate every Story, Spike, Refactor, and NFR
- Attendees are the Team and its stakeholders



Iteration Review guidelines

- Timebox: 1 – 2 hours
- Review preparation should be limited to 1 – 2 hours. Minimize PowerPoint. Work from the repository of Stories.
- If a major stakeholder cannot attend, the Product Owner should follow up individually.



Role	Feature	
Support	Enable Advanced Currency Management via blacktab	
Admin	Create Effective Dated Exchange Rates (EDER)	February Sprint
Admin	API access	
End User	Apply EDER to Opportunities (Detail Pages)	
Admin	Manage (View, Edit, Delete) Effective Dated Exchange Rates	
End User	Apply EDER to Opportunity Reports and Workflow	March Sprint
End User	Apply EDER to Opportunity Products and Schedules	
End User	Apply EDER to Customizable Forecasting	
Admin	Ability to define Transaction Dates per object	Future
End User	Apply EDER to all standard objects	
End User	Apply EDER to all custom objects	

Tooling is often used to facilitate the demo

Sample Iteration Review Agenda

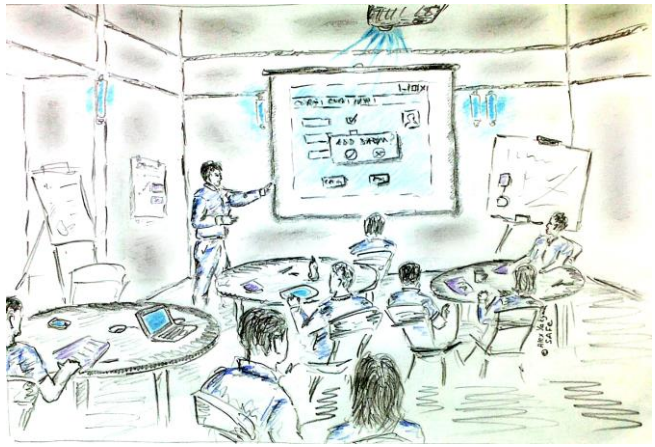
1. Review business context and Iteration Goals
2. Demo and feedback of each Story, spike, refactor, and NFR
3. Discussion of Stories not completed and why
4. Current risks and impediments
5. Revised team backlog and Team Pi Objectives as needed

The Iteration Review: Two views

The Iteration Review provides two views into the program:

1. How we did on the Iteration

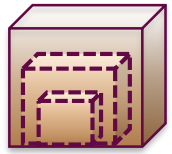
- Did we meet the goals?
- Story by Story review

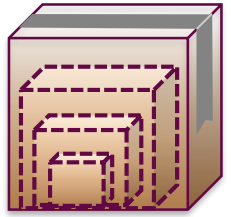


2. How we're doing on the PI

- Review of PI objectives
- Review remaining PI scope and reprioritize if necessary

SAFe Definition of Done





Team Increment	System Increment	Solution Increment	Release
• Stories satisfy acceptance criteria • Acceptance tests passed (automated where practical) • Unit and component tests coded, passed, and included in the BVT • Cumulative unit tests passed • Assets are under version control • Engineering standards followed • NFRs met • No must-fix defects • Stories accepted by Product Owner	• Stories completed by all teams in the ART and integrated • Completed features meet acceptance criteria • NFRs met • No must-fix defects • Verification and validation of key scenarios • Included in build definition and deployment process • Increment demonstrated, feedback achieved • Accepted by Product Management	• Capabilities completed by all trains and meet acceptance criteria • Deployed/installed in the staging environment • NFRs met • System end-to-end integration, verification, and validation done • No must-fix defects • Included in build definition and deployment/transition process • Documentation updated • Solution demonstrated, feedback achieved • Accepted by Solution Management	• All capabilities done and meet acceptance criteria • End-to-end integration and solution V&V done • Regression testing done • NFRs met • No must-fix defects • Release documentation complete • All standards met • Approved by Solution and Release Management

Exercise: Define your Definition of Done

- Consider the team increment Definition of Done in the previous slide, then create your own Definition of Done for Stories
- As a team, build a definition of what it means to you to finish a Story
- Be ready to present to the class

PREPARE

SHARE

4.7 Retrospect and improve

Iteration Retrospective

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
—Agile Manifesto

- 30 – 60 minutes
- Pick 1 – 2 things that can be done better, target for next Iteration
- Enter improvement items into the Team Backlog

Sample Agenda

Part 1: Quantitative

1. Review the improvement backlog items targeted for this Iteration. Were they all accomplished?
2. Did the team meet the goals (yes/no)?
3. Collect and review the agreed-to Iteration print metrics.

Part 2: Qualitative

1. What went well?
2. What didn't?
3. What we can do better next time?

Iteration Metrics

Functionality	Iteration 1	Iteration 2
# Stories (loaded at beginning of Iteration)		
# accepted Stories (defined, built, tested, and accepted)		
% accepted		
# not accepted (not achieved within the Iteration)		
# pushed to next Iteration (rescheduled in next Iteration)		
# not accepted: deferred to later date		
# not accepted: deleted from backlog		
# added (during Iteration; should typically be 0)		

Quality and test automation
% SC with test available/test automated
Defect count at start of Iteration
Defect count at end of Iteration
 # new test cases
 # new test cases automated
 # new manual test cases
Total automated tests
Total manual tests
% tests automated
Unit test coverage percentage

Exercise: Run a short retrospective

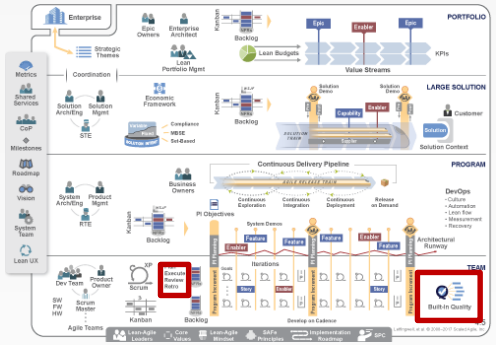
As a group, let's retrospect the course so far

- What went well?
- What didn't go so well?
- What can we improve for the next time we run this course?

Lesson summary

In this lesson, you:

- Defined and visualized the initial flow of work with your team
- Explored how to measure the flow of work
- Recognized techniques to build quality into development process
- Discussed how to continuously integrate, deploy and release value
- Explored how to demo value to team stakeholders
- Practiced running retrospectives



Suggested Scaled Agile Framework reading:

- *"Built-in Quality" article*
- *"Iteration Execution" article*
- *"Iteration Review" article*
- *"Iteration Retrospective" article*

Lesson 5
Executing the PI

1. Introducing the Scaled Agile Framework
2. Building an Agile Team
3. Planning the Iteration
4. Executing the Iteration
5. Executing the PI

SAFe® Course Attending this course gives students access to the SAFe Practitioner exam and related preparation materials.

Learning objectives

5.1 Plan together

5.2 Integrate and demonstrate together

5.3 Learn together

5.1 Plan together

PI Planning

Cadence-based PI Planning meetings are the pacemaker of the Agile Enterprise.

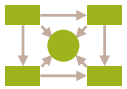
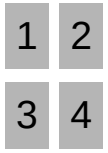
- Two days every 8 – 12 weeks (10 weeks typical)
- Everyone attends in person if at all possible
- Product Management owns Feature priorities
- Development teams own story-planning and high-level estimates
- Architects/Engineering and UX work as intermediaries for governance, interfaces, and dependencies

The goal of PI Planning

Alignment to a common mission!

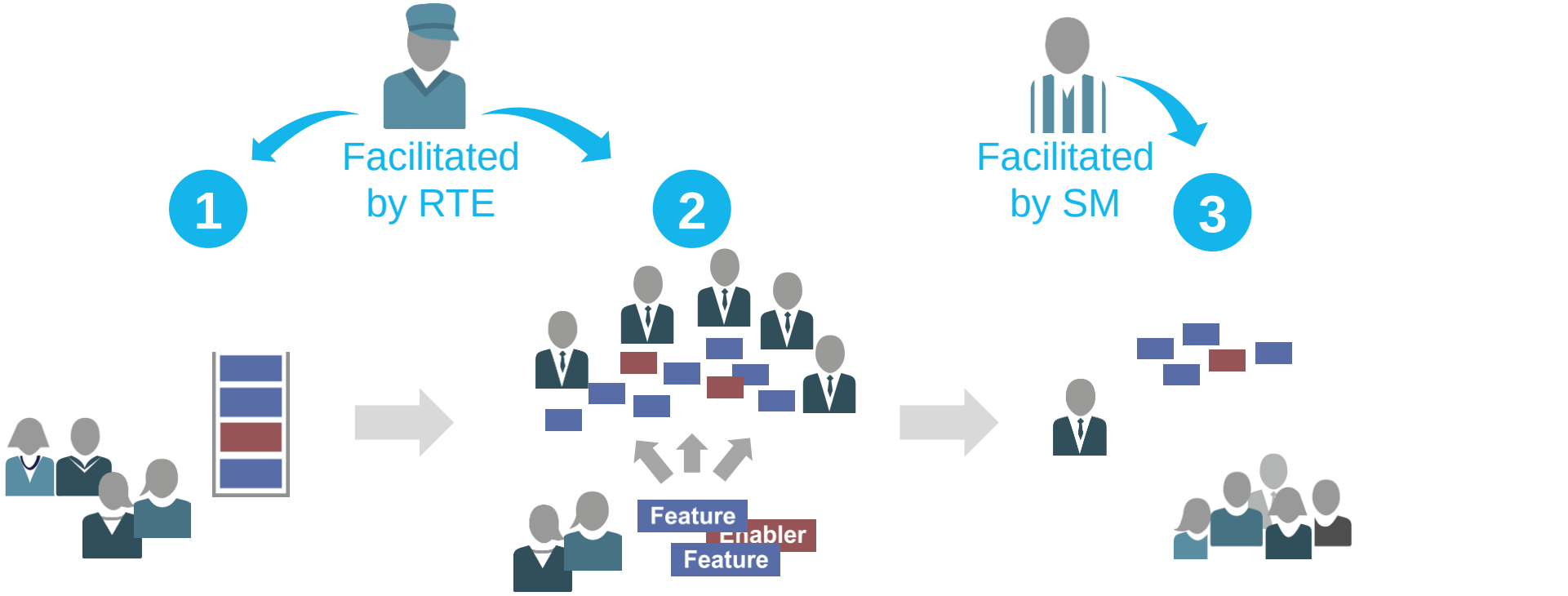
We are here to gain alignment and commitment around a clear set of prioritized objectives. I will now review the agenda for the next two days of the PI Planning event.

Day 1 agenda

8:00-9:00	Business context		State of the business and upcoming objectives
9:00-10:30	Product/Solution Vision		Vision and prioritized features
10:30-11:30	Architecture Vision & development practices		▸ Architecture, common frameworks, etc. ▸ Agile tooling, engineering practices, etc.
11:30-1:00	Planning context and lunch		Facilitator explains planning process
1:00-4:00	Team breakouts		▸ Teams develop draft plans and identify risks and impediments ▸ Architects and Product Managers circulate
4:00-5:00	Draft plan review		Teams present draft plans, risks, and impediments
5:00-6:00	Management review & problem solving		Adjustments made based on challenges, risks, and impediments

New PI content should not be a surprise

Upfront presentation of content to the teams solves a lot of problems later, during PI Planning.

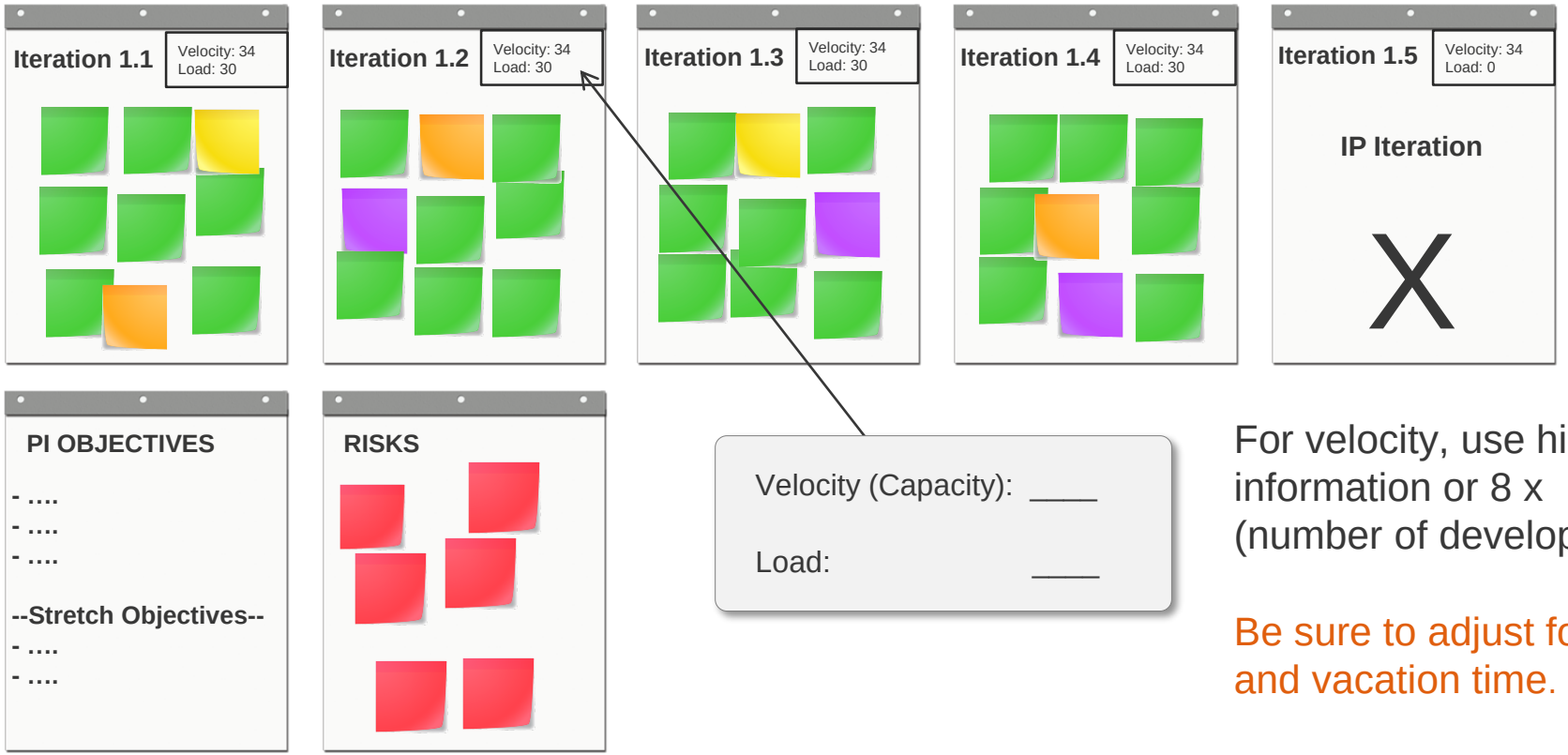


Product Management and other stakeholders refine Program Backlog

Features are roughly split into story-like chunks with POs

Each PO presents initial stories to the team, looks for feedback, big unknowns, etc.

Team breakout #1



For velocity, use historic information or 8 x (number of developers + testers).

Be sure to adjust for holidays and vacation time.

Color coding gives visibility into investments

Green	= User stories	Yellow	= Exploration Enablers	Red/ Pink	= Risks and dependencies
Orange	= Infrastructure/ Enablers	Purple	= Maintenance	Red/ Pink ✓	= Addressed risks and dependencies

Align to a mission with PI Objectives

Objectives are business summaries of what each team intends to deliver in the upcoming PI.

They often map directly to the features in the backlog … But not always.

For example:

- Aggregation of a set of Features, stated in more concise terms
- A Milestone like a trade show
- An Enabler Feature needed to support the implementation
- A major refactoring

Objectives for PI 1	**Business Value**
› Structured location and validation of locations	
› Build and demonstrate a proof of concept for context images	
› Implement negative triangulation by: tags, companies, and people	
› Speed up indexing by 50%	
› Index 1.2 B more web pages	
› Extract and build URL abstracts	
=== STRETCH ===	
› Fuzzy search by full name	
› Improve tag quality to 80% relevance	

Stretch objectives

Stretch objectives provide a reliability guard band.

Stretch objectives *do* count in velocity/capacity:

- They are planned, not extra things teams do “just in case you have time”
- But, they *are not included* in the commitment, thereby making the commitment more reliable
- If a team has low confidence in meeting a PI Objective, encourage them to move it to stretch
- If an item has many unknowns, consider moving it to stretch, and put in early spikes

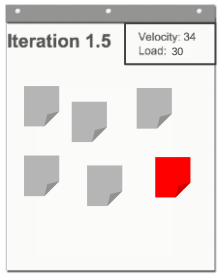
SMART Team PI Objectives

Teams should write their PI Objectives in the “SMART” format.

- **S**pecific States the intended outcome as simply, concisely, and explicitly as possible (Hint: Try starting with an action verb)
- **M**easurable It should be clear what a team needs to do to achieve the objective. The measures may be descriptive, yes/no, quantitative, or provide a range.
- **A**chievable Achieving the objective should be within the team’s control and influence
- **R**ealistic Recognize factors that cannot be controlled. (Hint: Avoid making “happy path” assumptions)
- **T**ime-bound The time period for achievement must be within the PI, and therefore all objectives must be scoped appropriately

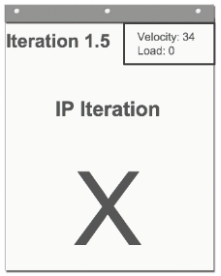
Team deliverables detail

Iterations



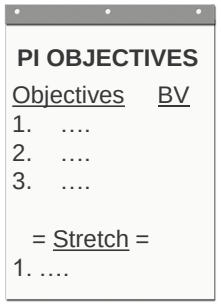
- If a Story has a dependency, put a red sticky on it describing the dependency. Put a check mark through it once the dependency has been addressed.
- If a risk is broader in nature, put it on the risk sheet
- If needed, allocate a percentage of capacity for unplanned activities like maintenance and production support

IP



- The last Iteration will be used for Innovation and Planning (IP)
- You should have a velocity but not a load on the IP Iteration, since it should not contain any user value stories

Objectives

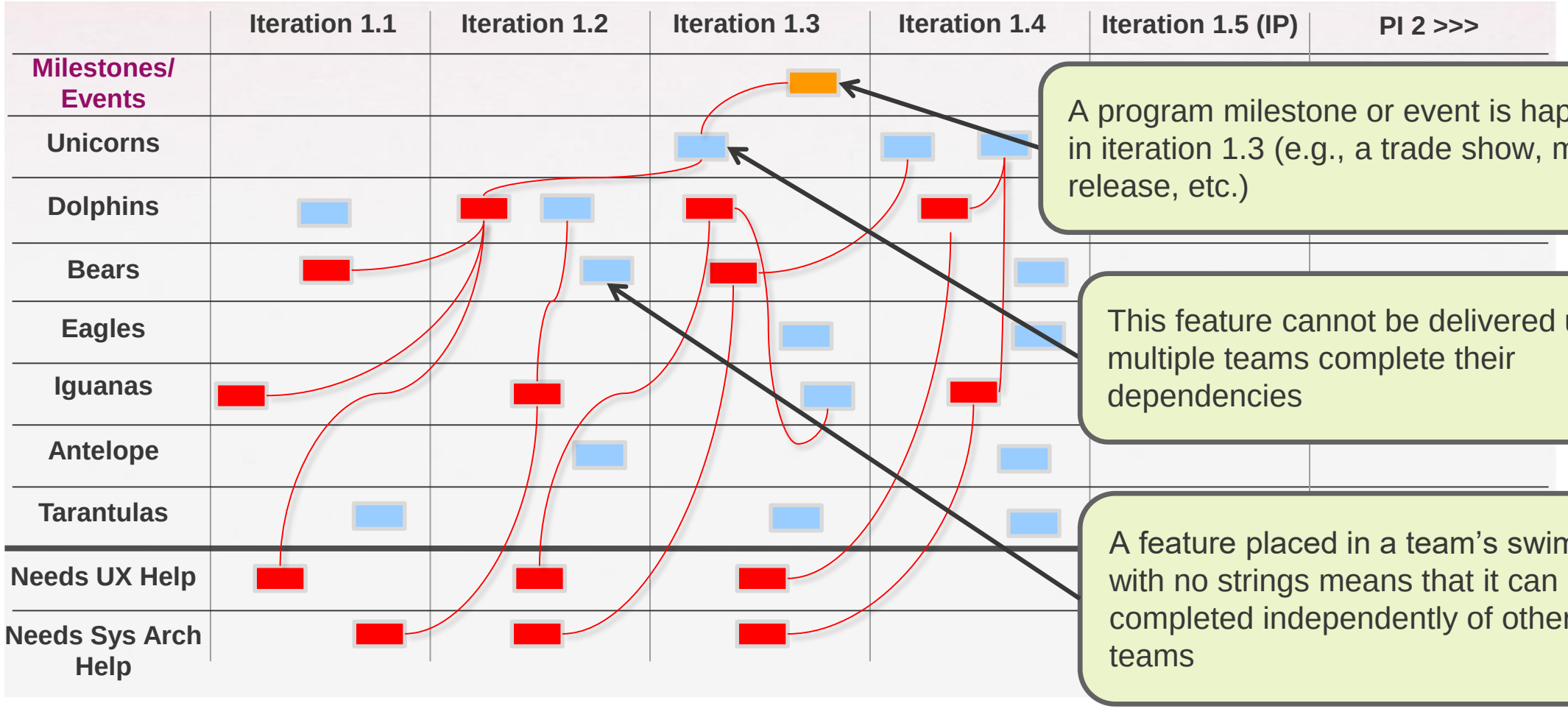


- PI Objectives should be written as “SMART” objectives
- Objectives are assigned business value during the second team breakout
- Stories supporting stretch objectives are included in the load calculation

Risks

- Program risks are those that need to be escalated to the program level. They will be captured and “ROAMed” after the final plan review.
- Team risks are those under the team’s control. They won’t be presented.

Program board - Feature delivery, dependencies and Milestones



Blue = Features

Red/ Pink = Significant Dependency

Orange = Milestone/ Event

Red String = A dependency requiring stories or other dependencies to be completed before the feature can be completed

Exercise: Identify dependencies

In your team, review the Stories you created from your Features and identify dependencies with other teams.

- If you are sitting with your whole ART:
  - Identify which teams you are likely to be regularly dependent on
  - Identify which teams you think will regularly be dependent on you
- Create a list of dependencies for your Stories or Features and be ready to present the teams you depend on and that depend on you

Management review and problem-solving

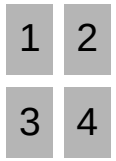
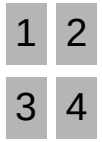
At Day 1 end, management meets to make adjustments to scope and objectives based on the day's planning.

Common questions during the manager's review:

- What did we just learn?
- Where do we need to adjust Vision? Scope? Resources?
- Where are the bottlenecks?
- What Features must be de-scoped?
- What decisions must we make between now and tomorrow to address these issues?

Used with permission of Hybris Software

Day 2 agenda

8:00-9:00	Planning adjustments		Planning adjustments made based on previous day's management meeting
9:00-11:00	Team breakouts		▸ Teams develop final plans and refine risks and impediments ▸ Business Owners circulate and assign business value to team objectives
11:00-1:00	Final plan review & lunch		Teams present final plans, risks, and impediments
1:00-2:00	Program risks		
2:00-2:15	PI confidence vote		
2:15-???	Plan rework if necessary		If necessary, planning continues until commitment is achieved
After commitment	Planning retrospective & moving forward		▸ Retrospective ▸ Moving Forward ▸ Final Instructions

Presented by RTE

Team breakout #2

Based on new knowledge (and a good night's sleep), teams work to create their final plans.

- In the second team breakout, Business Owners circulate and assign business value to PI Objectives from low (1) to high (10)
- Teams finalize the Program Increment plan
- Teams also consolidate program risks, impediments, and dependencies
- Stretch objectives provide the capacity and guard band needed to increase cadence-based delivery reliability

Objectives For PI 1	Bus. Value
- Structured locations and validation of locations	7
- Build and demonstrate a proof of concept for context images	8
- Implement negative triangulation by: tags, companies and people	8
- Speed up indexing by 50%	10
- Index 1.2 billion more web pages	10
- Extract and build URL abstracts	7
===== **Stretch Objectives** =====	=====
- Fuzzy search by full name	7
- Improve tag quality to 80% relevance	4

Addressing program risks

After all plans had been presented, remaining program risks and impediments are discussed and categorized.

ROAMing risks:

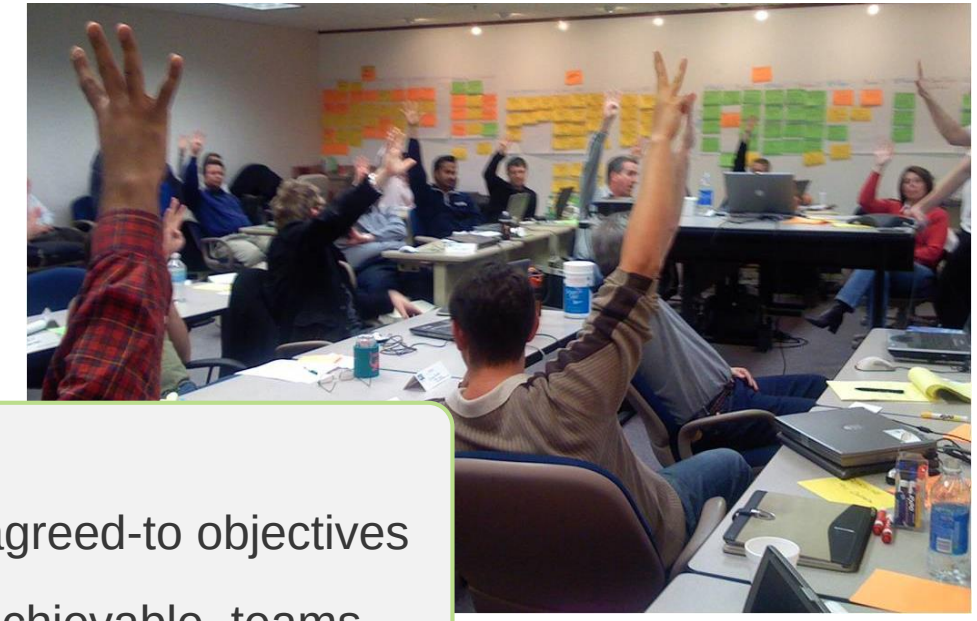
- **R**esolved – Has been addressed; no longer a concern
- **O**wned – Someone has taken responsibility
- **A**ccepted – Nothing more can be done. If risk occurs, PI may not yield the planned results.
- **M**itigated – Team has plan to adjust as necessary

Confidence vote: Team and Program Levels

After dependencies are resolved and risks are addressed, a confidence vote is taken at the Team and Program Levels.

"Fist of five" confidence vote

- Range of 1-5
- 1 = No confidence
- 5 = Very high confidence



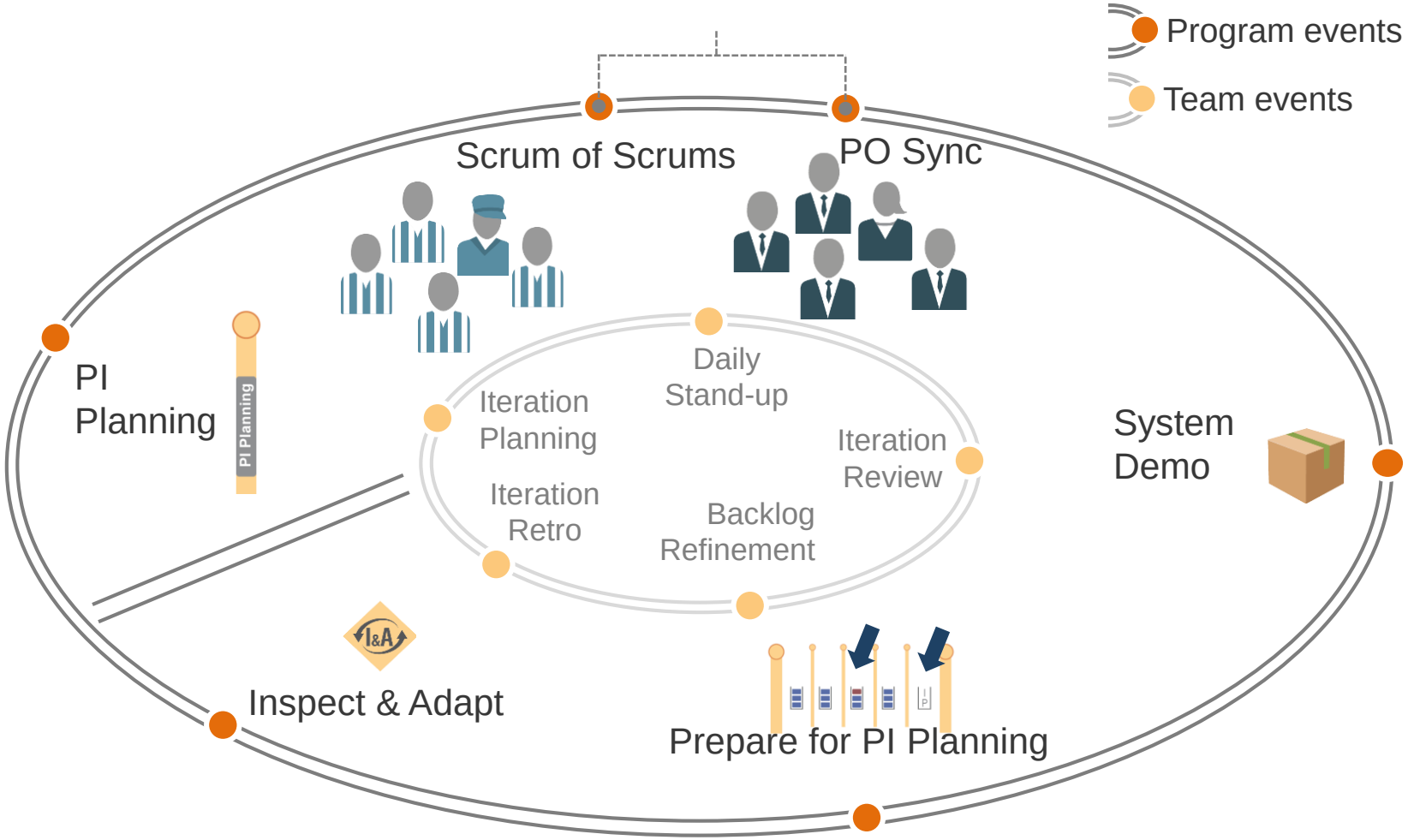
A commitment with two parts:

1. Teams agree to do everything in their power to meet the agreed-to objectives
2. In the event that fact patterns dictate that it is simply not achievable, teams agree to escalate *immediately* so that corrective action can be taken

5.2 Integrate and demonstrate together

Program execution

Program events create a closed loop system to keep the train on the tracks.



ART Sync

Programs coordinate dependencies through sync meetings.

ART Sync

Scrum of Scrums

- Visibility into progress and impediments
- Facilitated by RTE
- Participants: Scrum Masters, other select team members, SMEs if necessary
- Weekly or more frequently, 30 – 60 minutes
- Timeboxed, and followed by a “meet after”

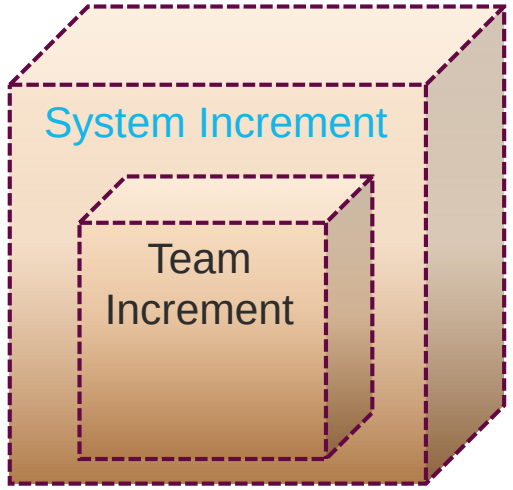
PO Sync

- Visibility into progress, scope, and priority adjustments
- Facilitated by RTE or PM
- Participants: PMs, POs, other stakeholders, and SMEs as necessary
- Weekly or more frequently, 30 – 60 minutes
- Timeboxed, and followed by a “meet after”

New system increment every two weeks

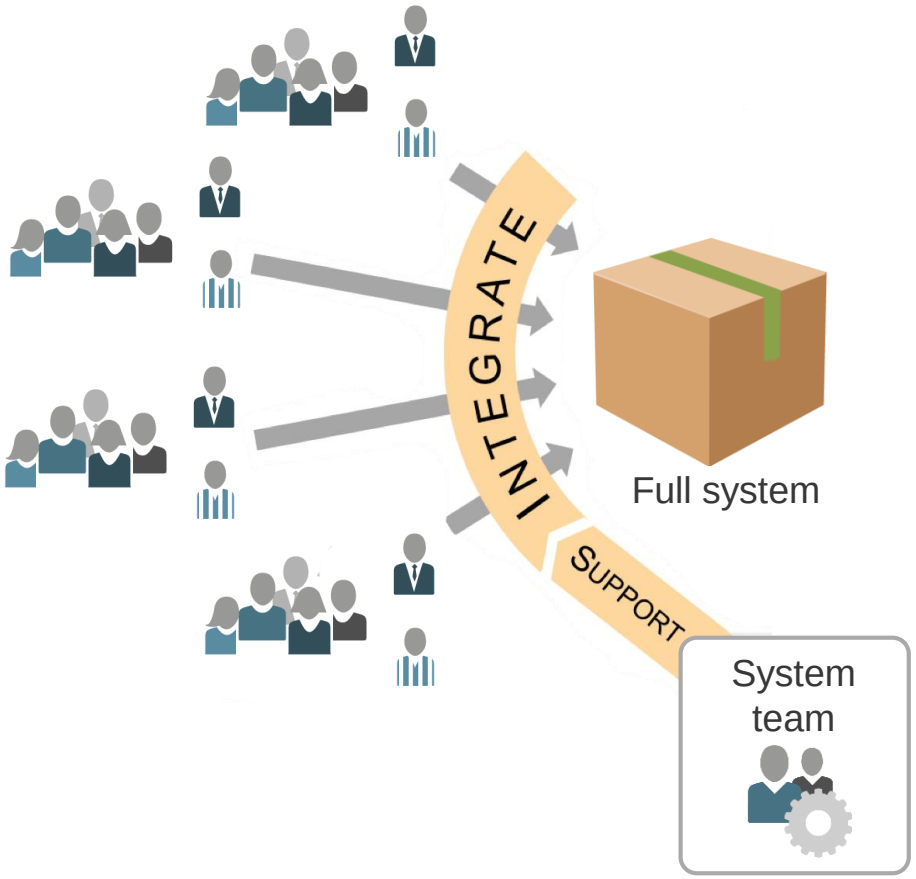
Every two weeks, teams evaluate the status of the new, integrated system increment.

- Features are functionally complete or “toggled” so as not to disrupt demonstrable functionality
- New Features work together, and with existing functionality
- Architectural Runway work in process is scaffolded and toggled
- System is continually verified via Story and Feature acceptance tests
- All practical NFR testing is done continuously



System Demo every two weeks

Demonstrate the full System increment to stakeholders every Iteration.



- An integrated System demo
- Happens after the iteration review (may lag by as much as one Iteration, maximum)
- Demo from the staging environment, or the nearest proxy

Exercise: System Demo challenges

- What are the challenges to having a new system increment every two weeks?
- Think about various aspects: environment, culture, tools and people
- In your group, prepare a list of 3 to 5 items that would make it hard to implement a System Demo of the integrated system every two weeks
- Be ready to present to the class

PREPARE

SHARE

5.3 Learn together

Innovation and Planning Iteration

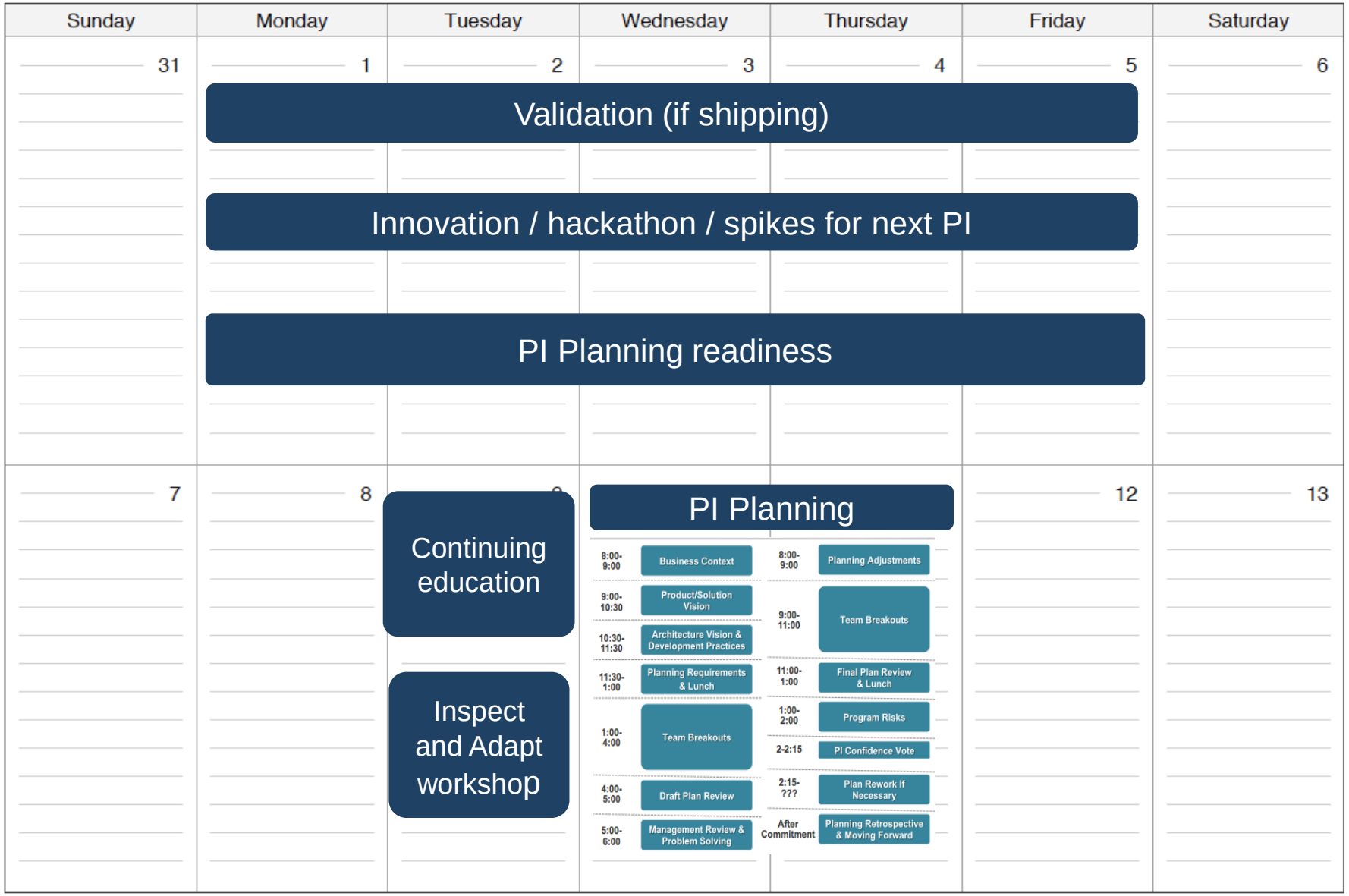
Facilitate reliability, Program Increment readiness, planning, and innovation

- Innovation: Opportunity for innovation spikes, hackathons, and infrastructure improvements
- Planning: Provides for cadence-based planning
- Estimating guard band for cadence-based delivery

Provide sufficient capacity margin to enable cadence.

—Don Reinertsen, Principles of Product Development Flow

IP Iteration calendar

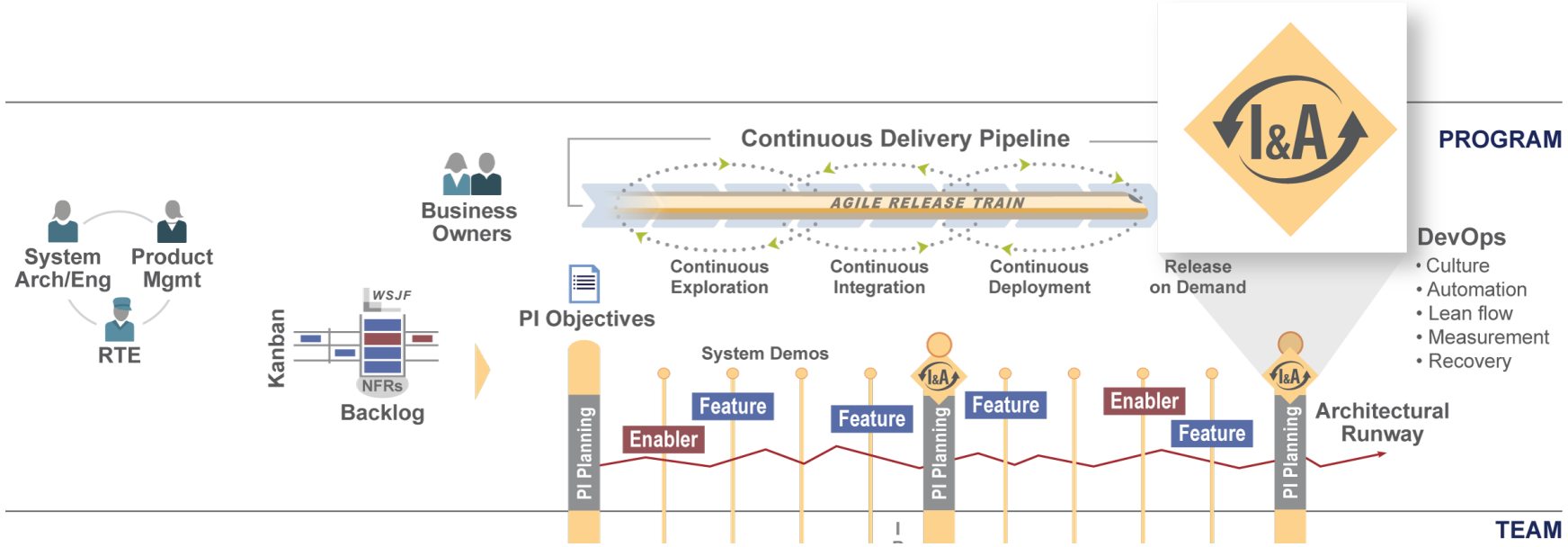


Inspect and Adapt

Three parts:

1. The PI System Demo
2. Quantitative measurement
3. The problem-solving workshop

- Attendees: Teams and stakeholders
- Timebox: 3 – 4 hours per PI



PI System Demo

At the end of the PI, teams demonstrate the current state of the Solution to the appropriate stakeholders.

- Often led by Product Manager
- Attended by Business Owners, program stakeholders, Product Management, RTE, Scrum Masters, and teams

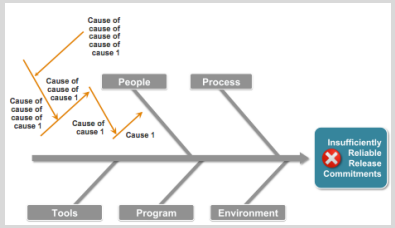
The problem-solving workshop

Teams conduct a short retrospective, then systematically address the larger impediments that are limiting velocity.

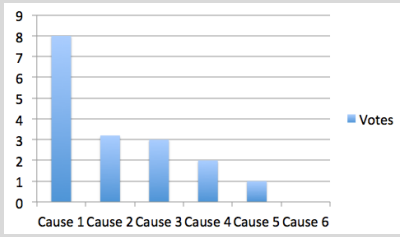
Agree on the problem to solve

Insufficiently reliable release commitments?

Apply root cause analysis (+ five whys)



Identify the biggest root cause using Pareto Analysis

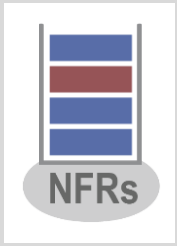


Restate the new problem for the biggest root cause

Insufficient Architectural Runway

Brainstorm solutions

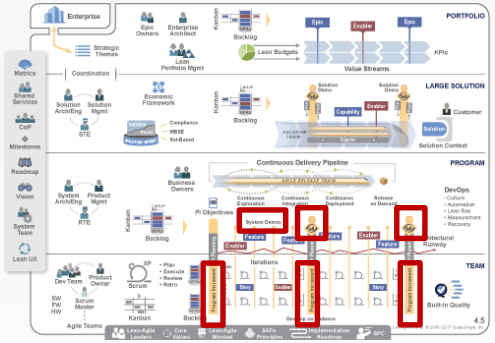
Identify improvement backlog items



Lesson summary

In this lesson, you:

- Learned how to plan and execute a Program Increment as a train
- Identified potential dependencies between teams
- Discussed the importance and challenges of the system demo
- Explored how to improve as a team of teams in the Inspect and Adapt workshop

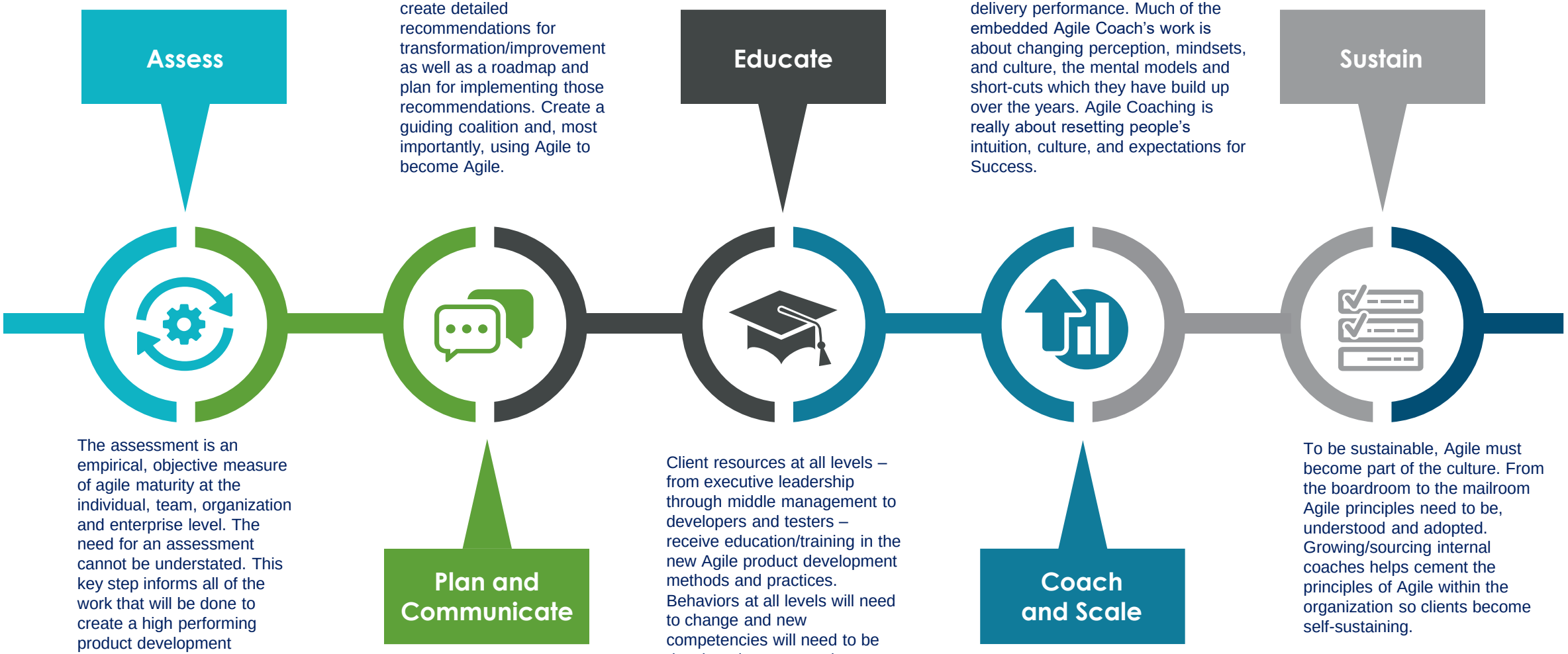


Suggested Scaled Agile Framework reading:

- *"Program Increment" article*
- *"System Demo" article*
- *"Inspect and Adapt" article*

Transformation Approach

ELIASSEN GROUP AGILE TRANSFORMATION MODEL



WATERFALL-A-HOLICS 12 STEPS

- Step 1: Kickoff - 1 day level and vision setting exercise with all effected parties from execs, business, product, tech dev, test, says test, test eng, Ops, etc.
- Step 2: Form Agile Strategic Team and create unifying narrative/vision (why are we transforming to Agile, Scrum and SAFe)
- Step 3: Get leadership trained - Leading SAFe/SAFe Fundamentals - and form Agile Executive Team
- Step 4: Identify initial DCDP work stream what we want as our initial/pilot work
- Step 5: Identify RTE and System Arch/Eng candidates and get them trained
- Step 6: Identify Product Manager/Product Owner candidates, vett, select, train and get started with coach to get backlogs created/refined
- Step 7: Identify Initial Scrum Master candidates, vett, select and train
- Step 8: Define initial set of teams - initial ART
- Step 9: Conduct SAFe for Teams for initial ART teams
- Step 10: Onboard coache(s) and intro to teams, begin coaching/working with backlogs
- Step 11: PI Pre-planning activities
- Step 12: PI Planning


Be the change you wish
to see in the world.
Mahatma Gandhi


SUCCESS IS A JOURNEY,
NOT A DESTINATION


JUST ASK
FIRE AWAY
QUESTION EVERYTHING
INQUIRY
GOT A
QUESTION
FOR ME?
WHO? WHAT?
WHERE? WHEN?
HOW? WHY?
EVERYTHING
BEGINS..
WITH
REALLY?
ANY
NO
YES
QUESTION
HOW MANY WORDS?
FRIENDS FORVER?
ISN'T
GREAT?
THAT COOL?
WHAT'RE
THEY TALKING
ABOUT
JUST DO IT?
ARE YOU
KIDDING ME?

Contact Info

Daniel Spencer

Enterprise Agile Coach

Eliassen Group Agile Practice

dspencer@eliassen.com

760.803.4227