Lesson 6
Executing and Releasing Value

Day 1

1. Introducing the Scaled Agile Framework
2. Embracing a Lean-Agile Mindset

   Break
3. Understanding SAFe Principles

   Lunch
4. Implementing an Agile Release Train

   Break
5. Experiencing PI Planning

Day 2

6. Executing and Releasing Value

   Break
7. Building an Agile Portfolio

   Lunch
8. Building Really Big Systems

   Break
9. Leading the Lean-Agile Enterprise

Learning objectives

6.1 Develop the Vision, Roadmap and Program Backlog

6.2 Prioritize the Program Backlog

6.3 Execute the Iterations in a Program Increment

6.4 Execute the Program Increment

6.5 Improve program performance with Inspect and Adapt

6.6 Release value on demand

Note: This lesson contains optional material in the Appendix

6.1 Develop the Vision, Roadmap and Program Backlog

Vision inspires action

It provides a longer term context:

- How will our future solution solve the larger customer problems?
- How will it differentiate us?
- What is the future context that our solutions will operate?
- What is our current business context, and how must we evolve to meet this future state?

Example: John Deer Vision
http://tinyurl.com/p5uloc5
25:38

Vision: a postcard from the future

- Aspirational, yet realistic and achievable
- Motivational enough to engage others on the journey

Result: The teams start thinking about how to apply their strengths in order to get there

Switch: How to Change Things When Change Is Hard, Chip Heath and Dan Heath, Broadway Books, 2010

Prepare the Solution Vision

The Solution Vision communicates strategic intent.

- Where are we headed with this Solution?
- What problem does it solve?
- What *features* and *benefits* does it provide?
- For whom does it provide them?
- What nonfunctional requirements (performance, reliability, platforms, etc.) does the solution deliver?

Common formats:

- Rolling wave briefings
- Vision document
- Preliminary data sheet
- Draft press release

Roadmap

The Roadmap guides the delivery of Features over time.

PI 3

- Road Rage ported (part I)
- Brickyard port started
- Distributed platform demo
- ALL GUIs for both games demonstrable
- Multiuser architecture
- New Road Rage features (see objectives for details)
- New Brickyard features (see objectives for details)

— Stretch Objectives —

- Demo of Beemer game

PI 4

- E3 Expo Tradeshow!
- Road Rage completed (single user)
- Brickyard Ported (single user)
- Road Rage multiuser demonstrable
- First multiuser game feature for Road Rage

PI 5

- Road Rage (multiuser) first release
- Brickyard ported multiuser demo
- New features for both games (see backlog)

Committed — *Forecast*

Exercise: Is a Roadmap a queue?

1. When, if ever, is a Roadmap a queue?
2. If the Roadmap on the prior slide was full, and committed, how long would it take to deliver a new, unanticipated feature?

Acceptance criteria

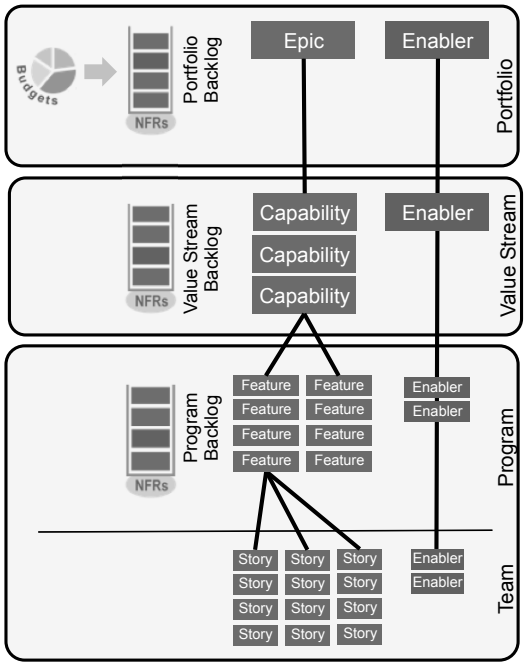
- Discuss at your table

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
- Fits in a PI

Feature	Benefit
Multiplexing	One PingOne connection enables secure, convenient access to multiple cloud applications.
Standards-based	Support for OAuth, SAML 2.0, OpenID and SCIM eliminates the risks of storing passwords or managing duplicate accounts in the Cloud.
Multi-factor Authentication	Support for authenticating a user by a password and a device (i.e., laptop, mobile device, etc.) for enhanced security.
Just-in-Time Provisioning	Add, delete or change users in your directory and their access is automatically updated across all supported cloud apps for centralized control. Supports the SCIM standard for user management.
CloudDesktop	Customizable at both the IT administrator and user levels, this portal provides a single point of access to all SSO-enabled private and public applications.



Features have benefits and acceptance criteria

- Benefits justify Feature implementation cost, provide business perspective when making scope decisions
- Business benefits impact economic prioritization of the Feature
- Acceptance criteria typically defined during Program Backlog refinement
- Reflect functional and Nonfunctional Requirements

SSO example:

Multi-factor authentication

Business benefit

Enhance user security via both password and a device.

Acceptance criteria

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

Exercise: Program Backlog

Identify three features from your business context.

Acceptance criteria

- Three features (or personal backlog items) items from your context
- Describe each feature in a short phrase, with a statement of benefits

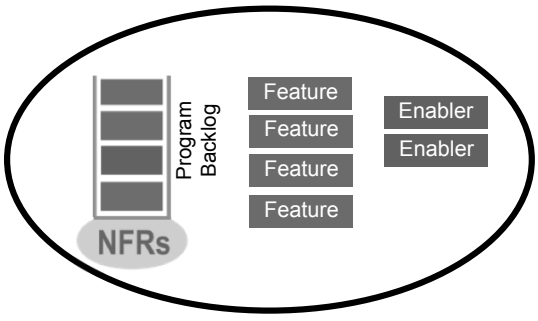
6.2 Prioritize the Program Backlog

Prioritize Features for optimal ROI

In a flow system, job sequencing is the key to economic outcomes.

To prioritize based on lean economics, we need to know two things:

1. What is the Cost of Delay (CoD) in delivering value?
2. What is the cost to implement the valuable thing?

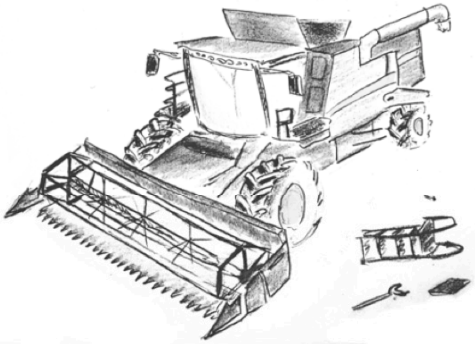


If you only quantify one thing, quantify the Cost of Delay.

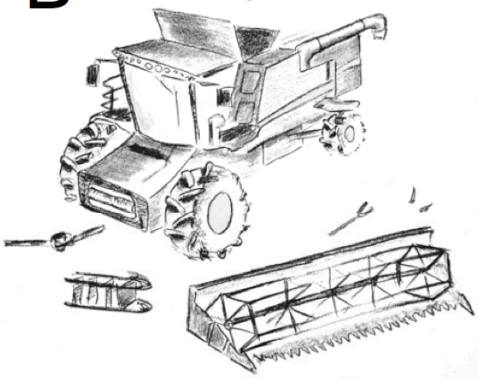
—Donald G. Reinertsen, Principles of Product Development Flow E3

Example with equal CoD: which job first?

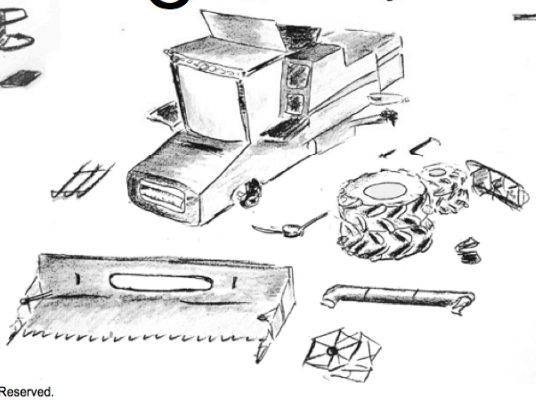
A $$, 1 day



B $$, 3 days

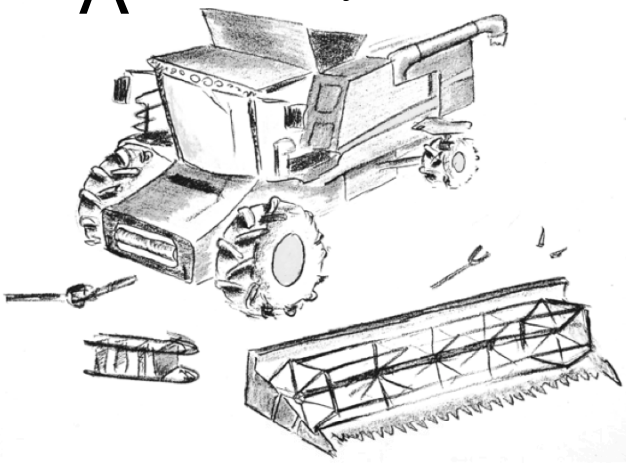


C $$, 10 days

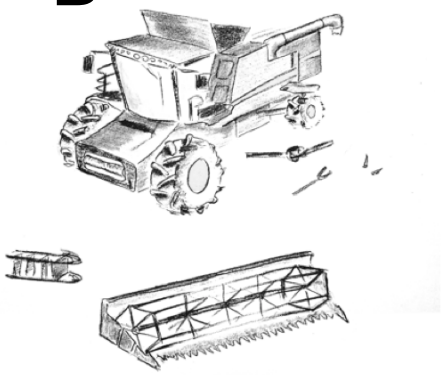


Example with equal Duration: which job first?

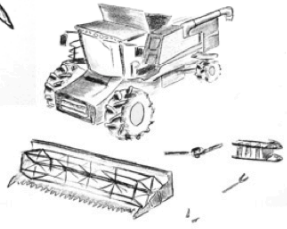
A $$$, 3 days



B $$, 3 days

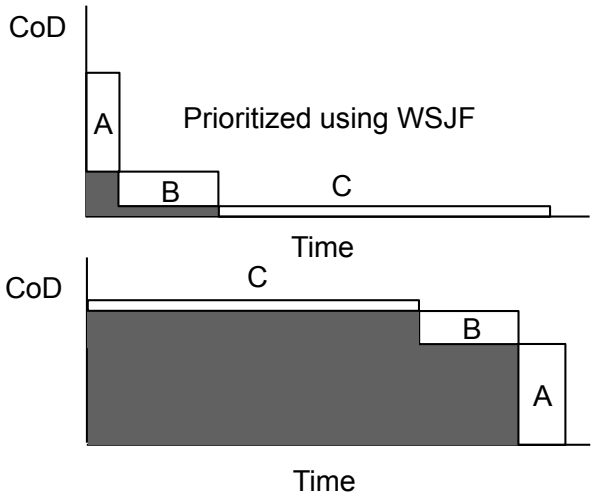


C $, 3 days



General case: any CoD and Duration

In the general case, give preference to jobs with shorter Duration and higher CoD, using *Weighted Shortest Job First* (WSJF):



$$WSJF = \frac{CoD}{Duration}$$

Feature	Duration	CoD	WSJF
A	1	10	10
B	3	3	1
C	10	1	0.1

– Dark area: total Cost of Delay

Adapted from *The Principles of Product Development Flow*, Donald G. Reinertsen

Components of Cost of Delay

User and business value	Relative value to the customer or business ▸ They prefer this over that ▸ Revenue impact? ▸ Potential penalty or other negative impact?
Time criticality	How User/Business Value decays over time ▸ Is there a fixed deadline? ▸ Will they wait for us or move to another solution? ▸ What is the current effect on customer satisfaction?
Risk Reduction & Opportunity Enablement (RR & OE)	What else does this do for our business ▸ Reduce the risk of this or future delivery? ▸ Is there value in the information we will receive? ▸ Enable new business opportunities?

Calculate WSJF with relative estimating

- In order to calculate WSJF, teams need to estimate Cost of Delay and duration
- For duration, use job size as a quick proxy for duration
- Relative estimating is a quick technique to estimate job size and relative value
- WSJF stakeholders: Business Owners, Product Managers, Product Owners, System Architects

$$\text{WSJF} = \frac{\text{CoD}}{\text{Job size}} = \frac{\text{User-business value} + \text{Time criticality} + \text{RR | OE value}}{\text{Job size}}$$

WSJF prioritization matrix

The job with the highest WSJF provides the greatest economic benefit.

$$\text{WSJF} = \frac{\text{CoD}}{\text{Job size}} = \frac{\text{User-business value} + \text{Time criticality} + \text{RR | OE value}}{\text{Job size}}$$

Feature	User-business value	Time criticality	RR | OE value	CoD	Job size	WSJF

Scale for each parameter: 1, 2, 3, 5, 8,13, 20
Note: Do one *column* at a time, start by picking the smallest item and giving it a "1."
There must be at least one "1" in each column!

Exercise: Prioritizing Program Backlog

Prioritize your backlog, based on cost of delay and job size

Instructions

- Using the prior page, prioritize your features using WSJF
- Estimate each feature one column (not row) at a time
- You will also have to estimate job size at this time

Acceptance criteria

- Three features are prioritized with WSJF

Exercise: Duration

- Question: Is job size always a good proxy for duration?
- When might that NOT be the case?
- How would you adjust based on that case?

6.3 Execute the Iterations in a Program Increment

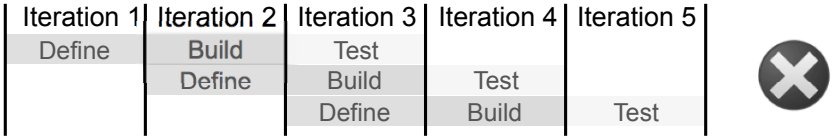
Note: this lesson's appendix contains optional exercise Iteration

Plan and commit

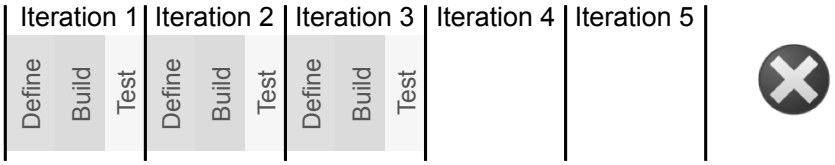
Purpose	Define and commit to what will be built in the iteration
Process	▸ The Product Owner defines *what* ▸ The Team defines *how* and *how much* ▸ Four hours max
Result	Iteration goals and backlog of the Team's commitment
Reciprocal commitment	▸ Team commits to delivering specific value ▸ Business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

Execution: Don't waterfall Iterations

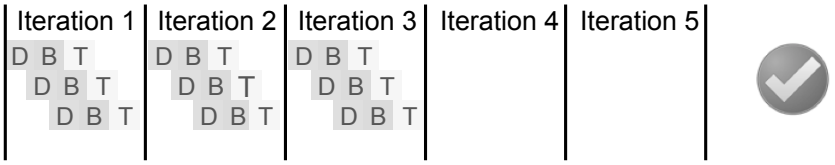
This is an inter-Iteration waterfall:



This is an intra-Iteration waterfall:



These are cross-functional Iterations:



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

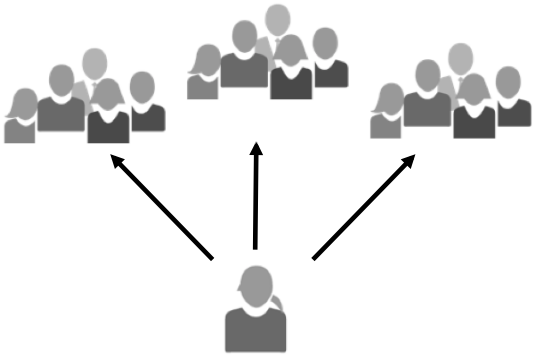
Built-in Quality

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 Architectural Runway

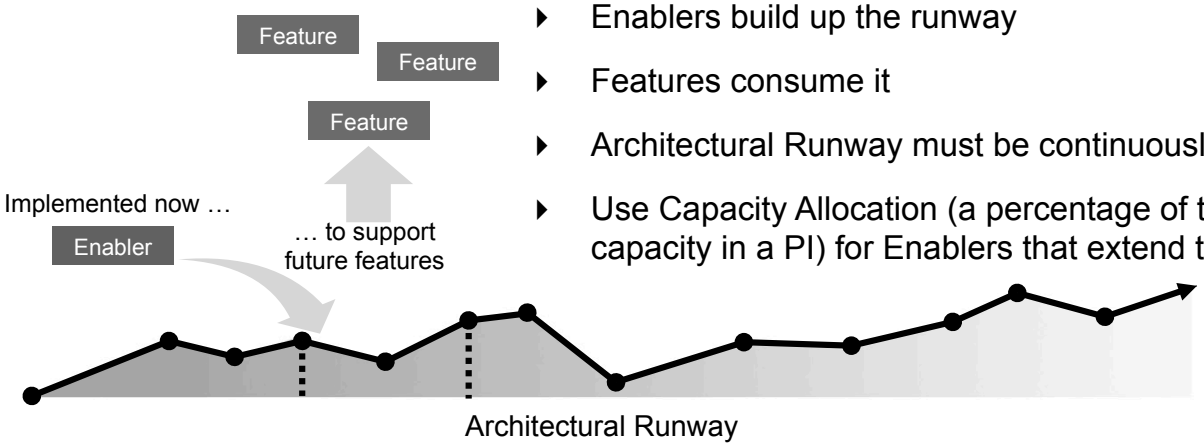


A balance between emergent design and intentional architecture is required for speed of development and maintainability

Architectural Runway

Architectural Runway—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

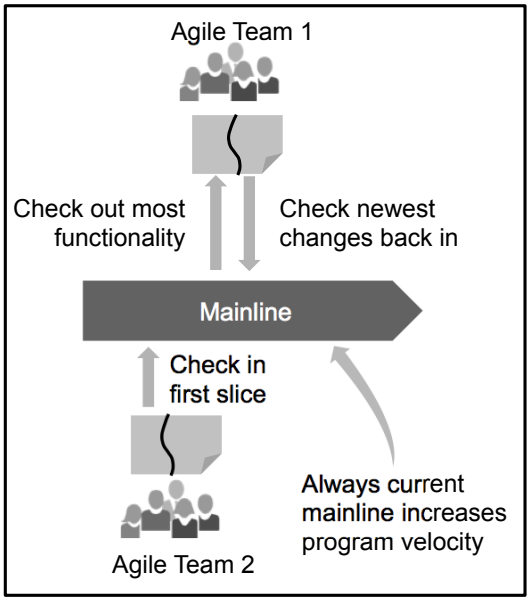


- Enablers build up the runway
- Features consume it
- Architectural Runway must be continuously maintained
- Use Capacity Allocation (a percentage of train's overall capacity in a PI) for Enablers that extend the runway

Continuous system integration

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

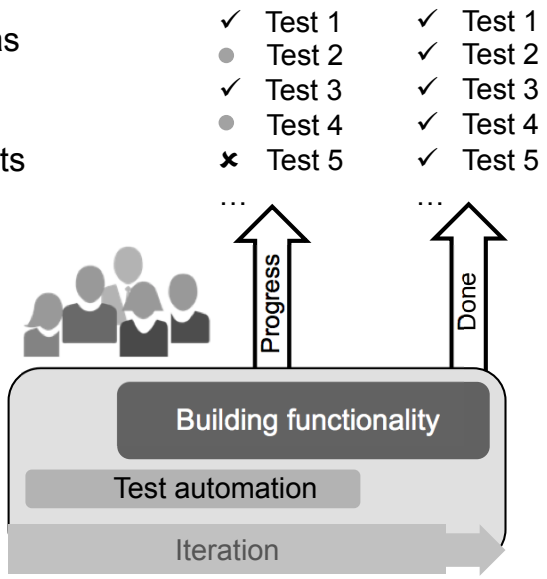
- Integrate every vertical slice of a user story
- Avoid physical branching for software
- Frequently integrate hardware branches
- Use development by intention in case of inter-team dependencies
  - Define interfaces and integrate first; then add functionality



Test first: Automate now!

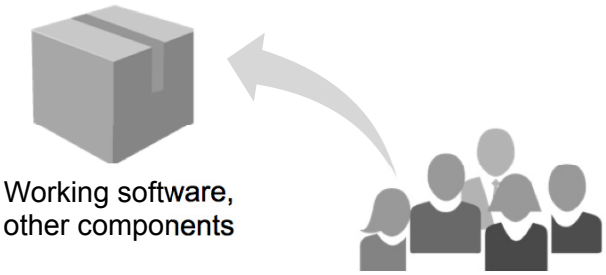
Else, velocity is bottlenecked, quality is speculative, scaling is impossible

- 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



The Team Iteration Demo

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



Note: This is the *Team* Demo, not *System* demo

Iteration Retrospective

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

Sample agenda

Part 1: Quantitative
1. Did the team meet the goals (yes/no)
2. Collect and review the agreed-to Iteration metrics

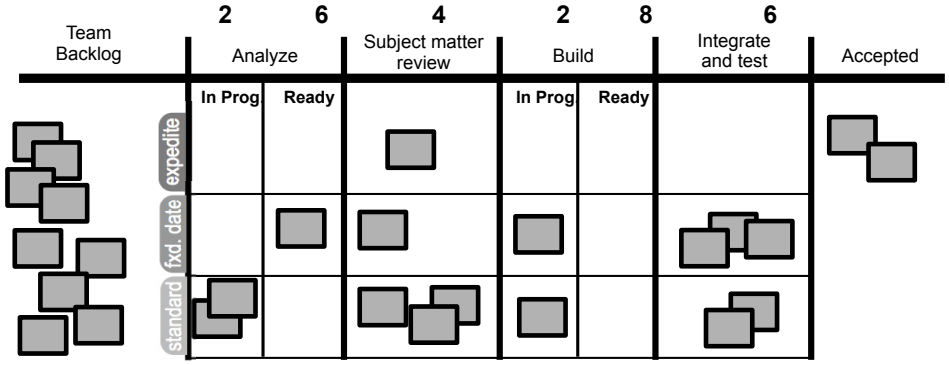
Part 2: Qualitative
1. What went well
2. What didn't
3. What we can do better next time

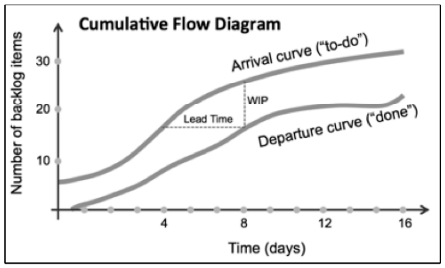
Guidelines

- 30–60 minutes
- Pick 1–2 things which can be done better, target for next Iteration

Use Kanban to improve flow

- Visualize workflow
- Establish WIP limits
- Use buffers (ex: “ready” above) to control variability
- Apply classes of service (e.g. “standard”, “fixed date”, “expedite”) to improve responsiveness
- Measure flow with Cumulative Flow Diagram (CFD)
- Continuously improve based on empirical flow data

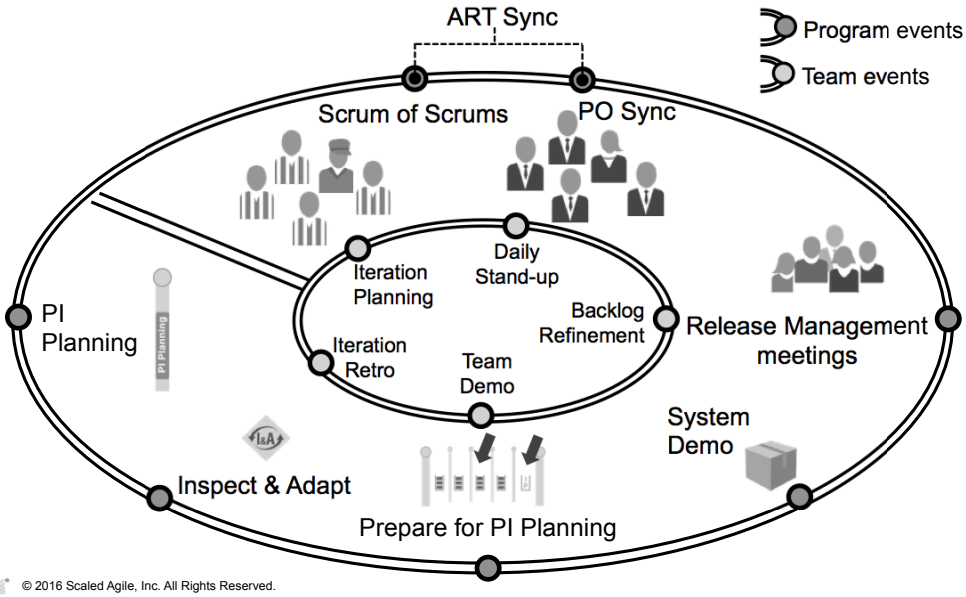




6.4 Execute the Program Increment

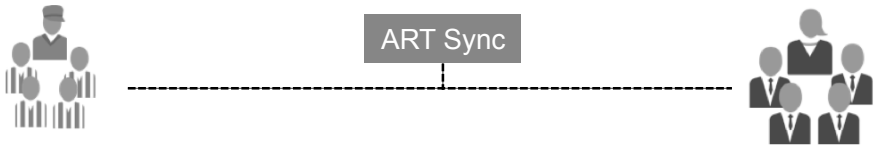
Program execution

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



ART Sync

Programs coordinate dependencies through sync meetings.



Scrum of Scrums

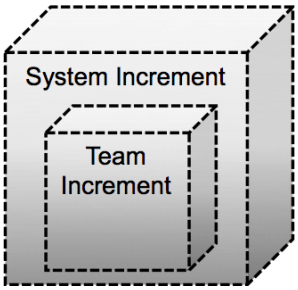
- Visibility into progress and impediments
- Facilitated by RTE
- Participants: Scrum Masters, other select team members, SMEs if necessary
- Weekly or more frequently, 30 – 60 minutes
- Timeboxed, and followed by a “meet after”

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

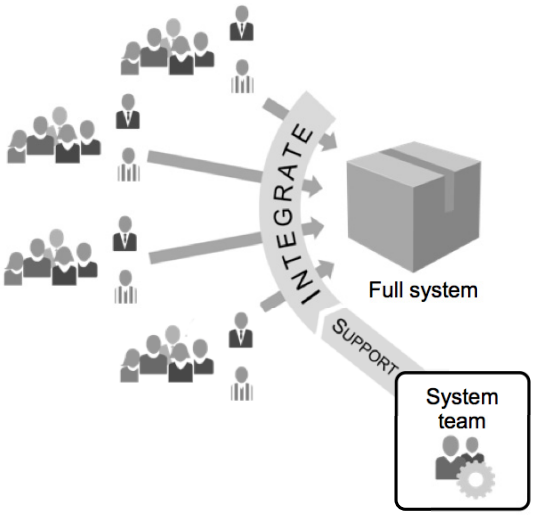
Every two weeks, teams evaluate the status of the new, integrated system increment.



- 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 Solution increment to stakeholders every Iteration.



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

Innovation and Planning Iteration

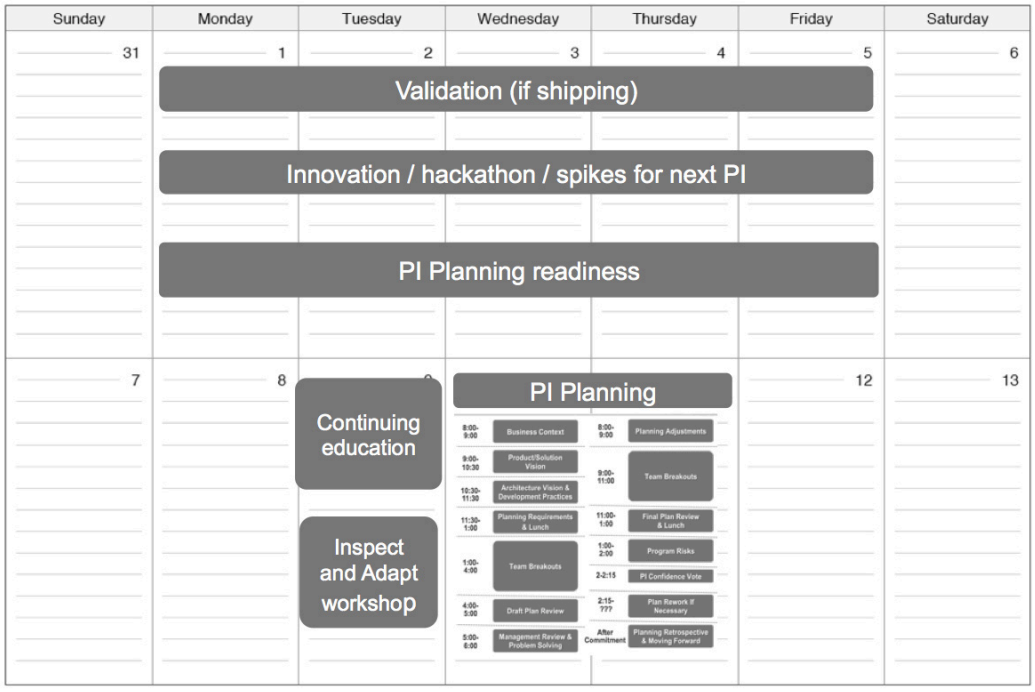
Facilitate reliability, Program Increment readiness, planning, and innovation

- 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



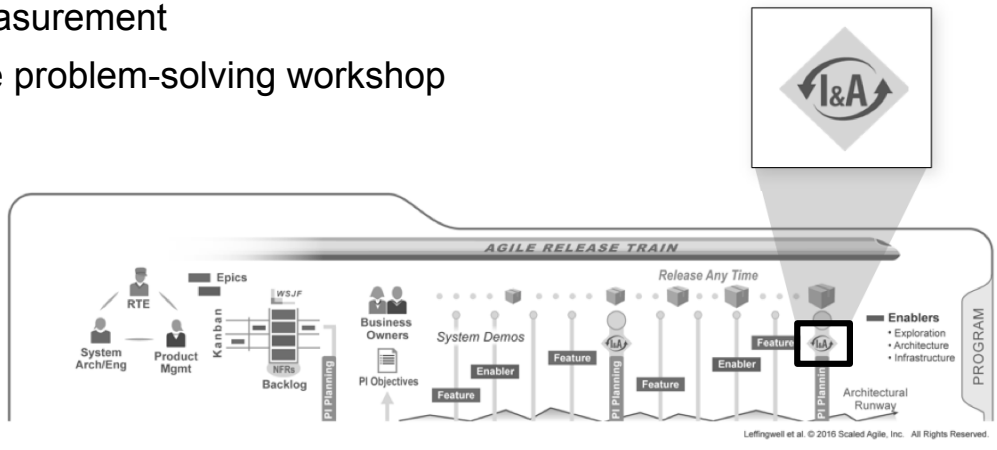
6.5 Improve program performance with Inspect and Adapt

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

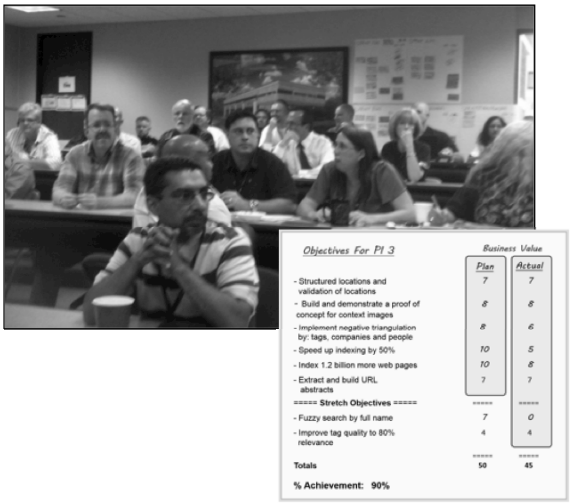
At the end of the PI, teams demonstrate the current state of the Solution to the appropriate stakeholders.

- Often led by Product Management, POs, and the System Team
- Attended by Business Owners, program stakeholders, Product Management, RTE, Scrum Masters, and teams

Program performance reporting

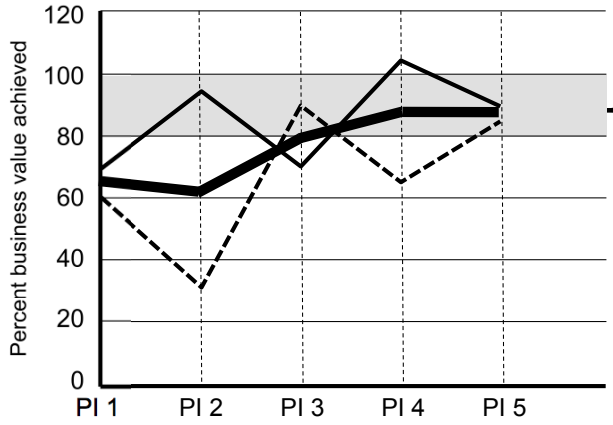
As part of the Solution Demo, teams compare planned vs. actual PI Objectives.

- Teams meet with their Business Owners to self-assess the business value they achieved for each objective
- Each team's planned vs. actual business value is then rolled up to the Program Level in the Program Predictability Measure



PI Predictability Measure

The PI Predictability Measure shows whether achievements fall into an acceptable process control band.

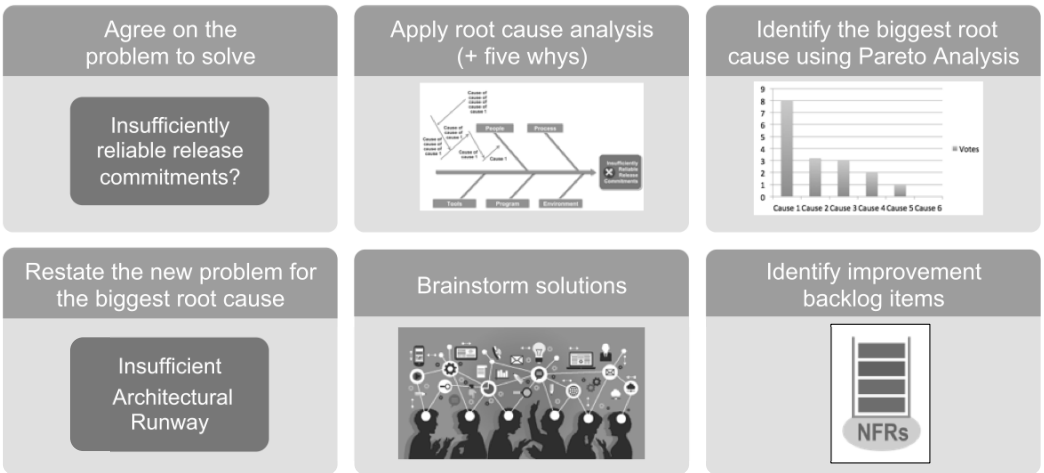


- Target: effective process control range
- Predictability sufficient to run the business
- Handles common variations
- Special causes may still cause excess variation

Team A (out-of-control development)
Team B (controlled development)
Program

The problem-solving workshop

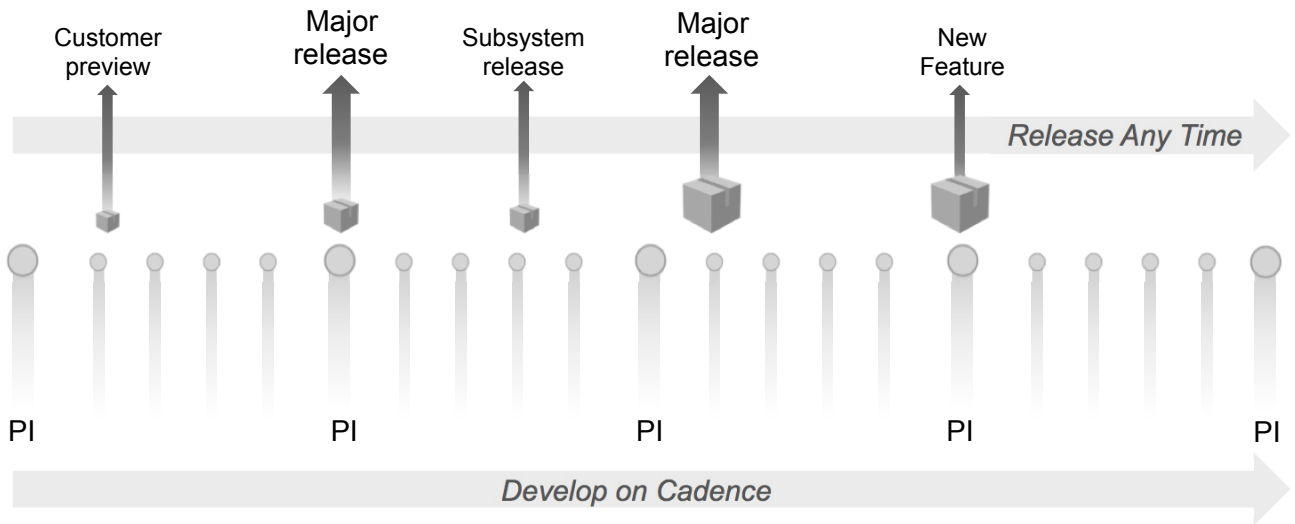
Teams conduct a short retrospective, then systematically address the larger impediments that are limiting velocity.



6.6 Release value on demand

Develop on Cadence. Release Any Time.

Development cadence limits variability to a single PI interval.
Release is on demand.



Build a deployment pipeline

Real value occurs only when the end users are successfully operating the Solution.

DevOps

Deployment pipeline streamlines delivery:

1. Staging environment emulates production
2. Development and test environments match production to the extent feasible
3. Working system is deployed to staging every Iteration
4. Supporting activities:
   - Everything under version control
   - Ability to automatically build environments
   - Automated the actual deployment process

Scott Prugh: DevOps in Legacy Environments
https://youtu.be/f4et0EGvKXA
25:38

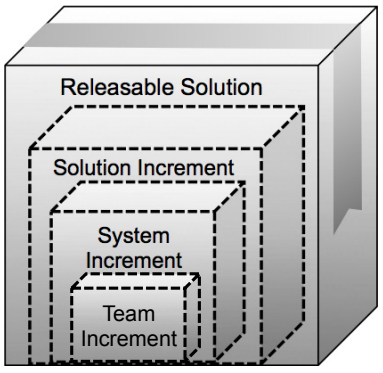
Releasing includes additional activities

System validation:

- User acceptance testing
- Final NFR testing
- Integration testing with other systems
- Regulatory standards and requirements

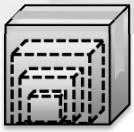
Documentation:

- Release communications
- End user documentation
- Bill of materials
- Training support personnel
- Installation/deployment instructions
- Legal, regulatory, other approvals
- etc. …



SAFe Definition of Done



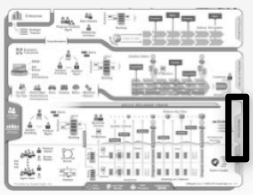



Team Increment	System Increment	Solution Increment	Release
• Stories satisfy acceptance criteria • Acceptance tests passed (automated where practical) • Unit and component tests coded, passed, and included in the BVT • Cumulative unit tests passed • Assets are under version control • Engineering standards 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

Lesson summary

In this lesson, you learned how to:

- Develop the Vision, Roadmap and Program Backlog
- Prioritize the Program Backlog
- Execute the Iterations in a Program Increment
- Execute the Program Increment
- Improve program performance with Inspect and Adapt
- Release value on demand



Suggested Scaled Agile Framework reading:
"Program Level" article

Appendix
Executing an Iteration: an exercise

Exercise: Iteration Planning

- Shortly, your team will execute a 20 minute Iteration exercise
- Using the exercise backlog in the appendix, plan how you will execute the Iteration
- Hint: Teammates *take responsibility* for stories

The goal: Plan to get as many stories (and points) accepted as you can during Iteration execution

Notes

- Ignore the prior team roles
- The trainer will select 2–3 persons to be Product Owners and will instruct them on their role for the upcoming "Iteration Execution" exercise. Therefore, they will not participate in Iteration Planning.
- All other persons at each table will participate in executing the Iteration

Exercise: Iteration Execution

- Execute the Iteration that you just planned
- The Product Owners will accept the backlog items

Instructions

- Your team has 20 minutes to complete as many backlog items as possible
- You can use any resources at your disposal that are not prohibited by the backlog item
- Only the product owner can accept and give credit for backlog items. This must happen during the Iteration time.

Acceptance criteria

- Team has some number of backlog items accepted by the PO
- Team must complete the **required** backlog items to get any credit

Exercise: Iteration Retrospective

Let's take a few minutes to understand what we would have done differently to increase our velocity.

Acceptance criteria

- Each team reflects on their results and has three suggestions for what they would do different next time
- Instructor will collect and summarize this data