Lesson 4

Facilitating Iteration Execution

1. Introducing Scrum in SAFe
2. Characterizing the role of the Scrum Master
3. Experiencing PI Planning
4. Facilitating Iteration Execution
5. Finishing the PI
6. Coaching the Agile Team

SAFe® Authorized Course: Attending this course gives learners access to the SAFe® Scrum Master exam and related preparation materials.

Learning objectives

4.1 Experience an Iteration

4.2 Plan the Iteration

4.3 Track the Iteration progress

4.4 Refine the Backlog

4.5 Facilitate the Iteration Review and System Demo

4.6 Continuously deploy with DevOps

4.7 Release on Demand

4.8 Facilitate relentless improvement

4.1 Experience an Iteration

Exercise: Iteration Planning

- Shortly, your team will execute a 15-minute Iteration exercise
- Using the exercise backlog in Appendix B, 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

PREPARE

Exercise: Iteration Planning (cont.)

- Ignore the prior team roles from the PI Planning simulation.
- The trainer will select 2 - 3 people 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.

PREPARE

Exercise: Iteration Execution

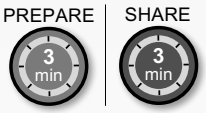
- Execute the Iteration that you just planned.
- The Product Owners will accept the backlog items.
- Your team has 15 minutes to complete as many backlog items as possible.
- You can use any resources at your disposal that are not prohibited by the backlog item.
- Only the Product Owner can accept and give credit for backlog items. This must happen during the Iteration time.

PREPARE

Exercise: Iteration Retrospective

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

- Each team should reflect on their results and has three suggestions for what they would do differently next time
- The trainer will collect and summarize this data



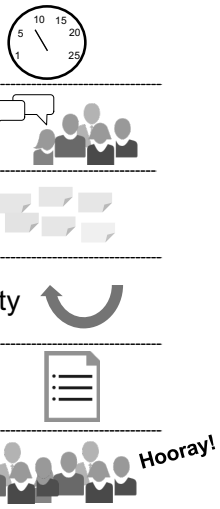
4.2 Plan the Iteration

Plan and commit

Purpose	Define and commit to Iteration Goals
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

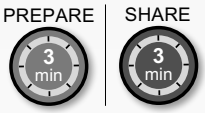
1. The team establishes its velocity
2. The team clarifies the Stories
3. The team optionally breaks Stories into tasks
4. The process continues while there is more capacity
5. The team synthesizes Iteration Goals
6. Everyone commits



- Timebox: 4 hours
- This meeting is by and for the team
- Subject matter experts (SMEs) may attend as required

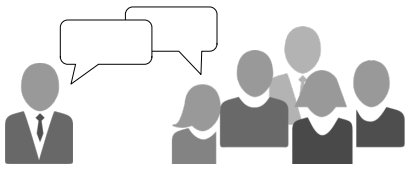
Exercise: Velocity

- Your team has a velocity of 20 Story points per Iteration. The group manager comments that another team on the train has a velocity of 40 Story points per Iteration.
- Discuss in your group how can you improve your velocity to surpass the other team.



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 defined and refined
  - Is estimated
- The process continues until the estimation of the Stories has reached the velocity of the team



Iteration Goals

Iteration Goals provide clarity, commitment, and management information.

They serve three purposes:

1. Align team members to a common purpose
2. Align Agile 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

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 the concern.

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.

Using tasks to plan the Iteration

- A common approach is to split Stories into tasks during Iteration Planning
  - Tasks should be under a day and are assigned to individuals
  - The capacity is calculated based on availability and not velocity

- This pattern, while common, is not officially part of Scrum and is not recommended in SAFe
- Tasks tend to draw the focus from achieving valuable Stories to checking off technical tasks
- They can reduce the overall team commitment to a Story in favor of a personal commitment to tasks
- However, while this pattern has problems, it is sometimes useful for beginning teams to ensure that their Iteration plans are sound

The Scrum Master's role in Iteration Planning

- Maintain timebox
- Ensure that the team commits to the Iteration Goals
- Verify that the PO or other managers don't influence the team to overcommit
- Challenge the team to exceed their previous accomplishments
- Ensure that improvement items from the retrospective are put into effect
- Ensure time is allocated for technical debt activities

Common anti-patterns

- Delving too deep into technical discussions
- Commitment is unrealistic
- Velocity and load are exactly the same
- Scrum Master is more focused on technical hat than facilitator's hat
- The team under-commits due to fear of failure
- No time is reserved for support activities

Exercise: Iteration Goals

For the upcoming Iteration, create some Iteration goals to reflect the following:

- Features, Feature slices, or Feature aspects, such as research, necessary infrastructure, etc.
- Business or technical Milestones
- Architectural, technical, or infrastructure effort
- Routine jobs
- Other commitments, such as maintenance, documentation, etc.

PREPARE 5 min | SHARE 2 min

4.3 Track the Iteration progress

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

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.

Caution!

Poor daily stand-ups may be a symptom of a deeper problem that requires a more systematic approach.

Potential root causes:

- Poor collaboration of the team members during the Iteration (e.g., Vijay does not know and doesn't care about what Ken is working on, and vice versa)
- Lack of collective code ownership
- Infrequent verification and integration during the Iteration (e.g., we are working on something, and we think it's good)
- Perpetual, unresolved conflict within the team

Exercise: Facilitating the Daily Stand-Up

- Your group is an Agile Team
- Choose a Scrum Master and have him read your team project
- Each person picks up a secret identity card. Don't show it to others!
- Run a Daily Stand-Up playing your role as dictated by the card
- If the Scrum Master calls you on your specific behavior, you can stop

PREPARE 5 min | SHARE 2 min

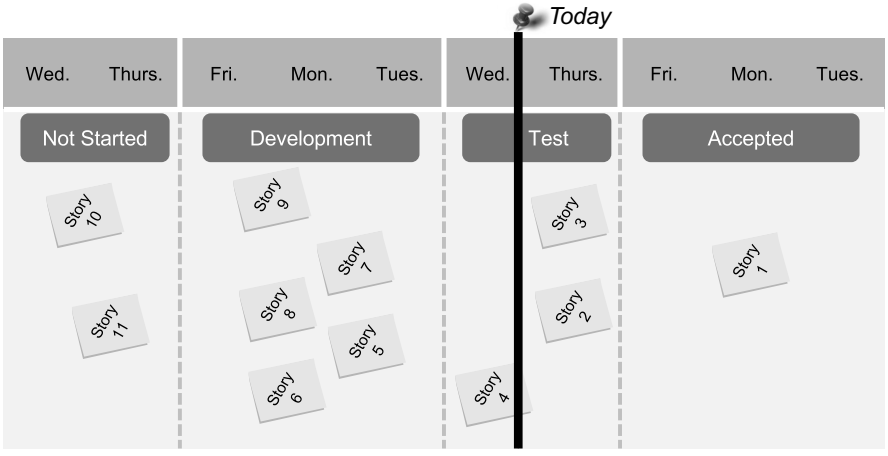
Exercise: Daily Stand-Up debrief

- Scrum Master, how was it for you?
- Team members, what insights do you have for the Scrum Master?
- How can we deal with those behaviors when they come up in Daily Stand-Up meetings?

SHARE

Using the team board to track progress

The board acts as a Big Visible Information Radiator (BVIR).

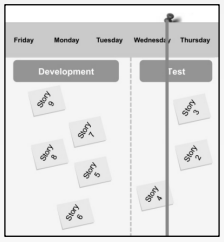


How is this team doing? How do you know that?

Exercise: Work-In-Process (WIP) constraints

Consider the BVIR on the previous page, then discuss:

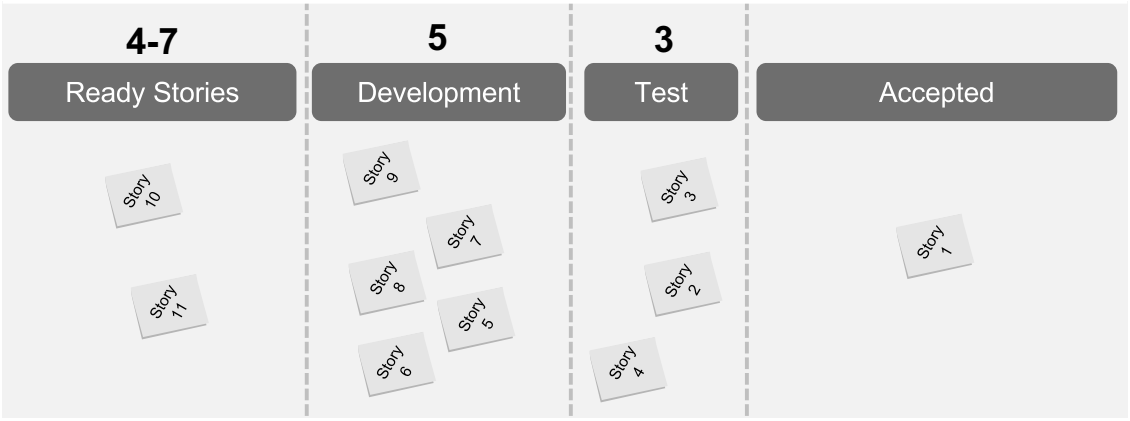
- What would the effect be of a 3-Story WIP constraint on development and testing?
- Scenario: You're a developer. You just finished Story 6. What would you do if:
  a. There is no WIP constraint?
  b. The 3-Story WIP constraint is in place?
- Which scenario has the highest throughput?



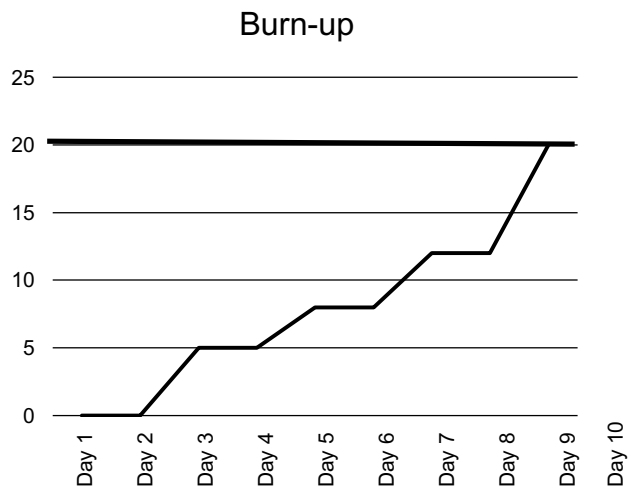
PREPARE 3 min | SHARE 3 min

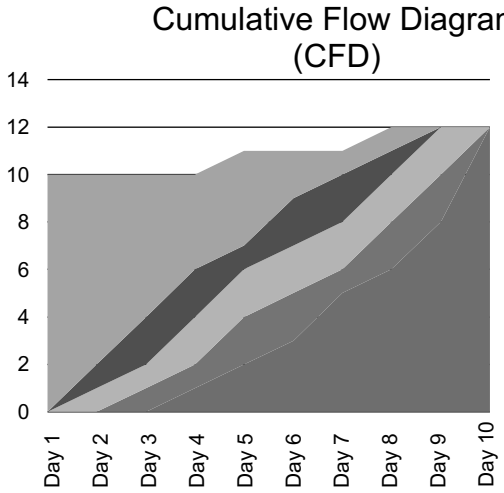
Setting WIP limits

WIP limits improve the flow of work. Some steps have no WIP limits, while others serve as buffers and have minimal as well as maximal WIP.

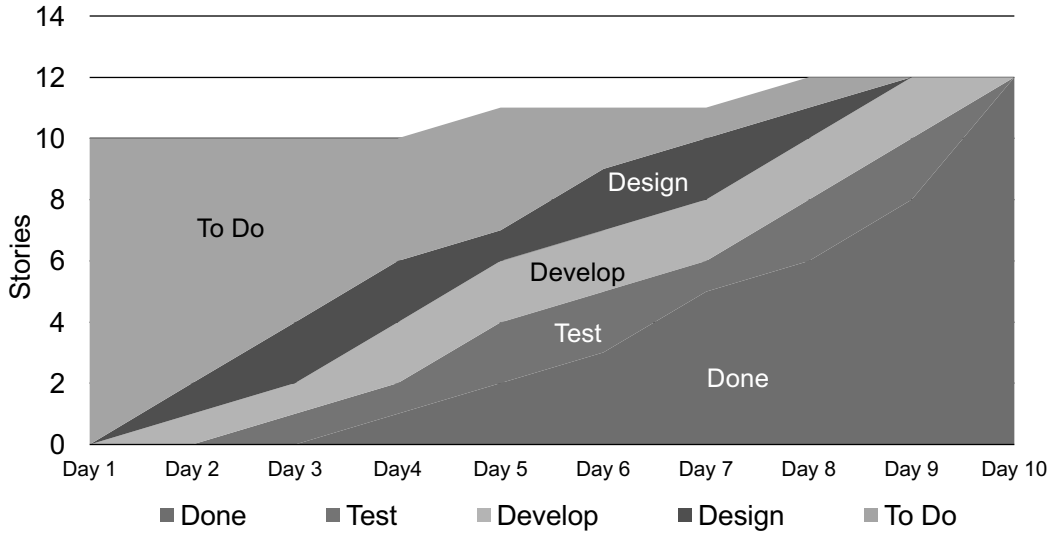


Track status with burn-up charts and CFDs

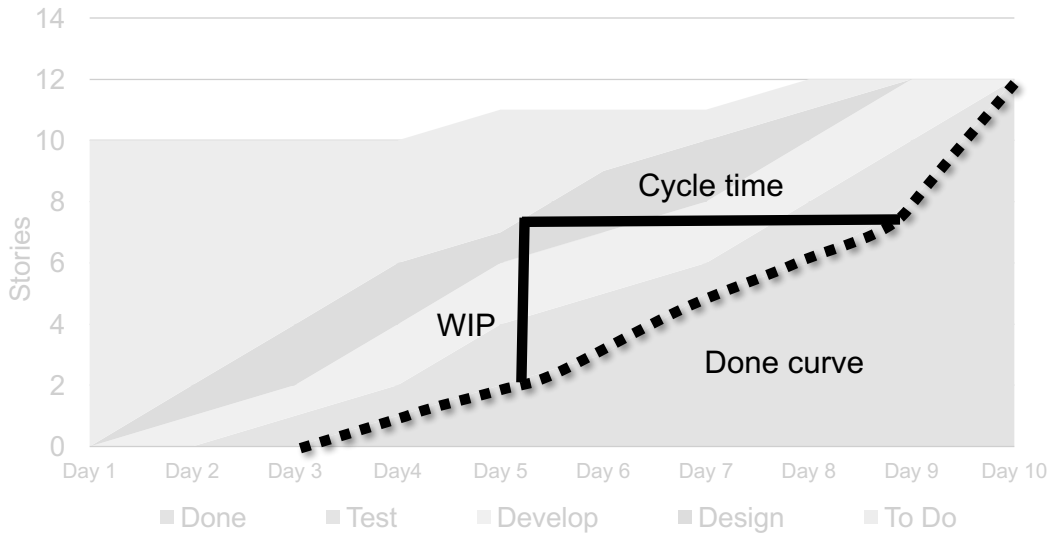




Understand CFDs



What can you learn from a CFD?

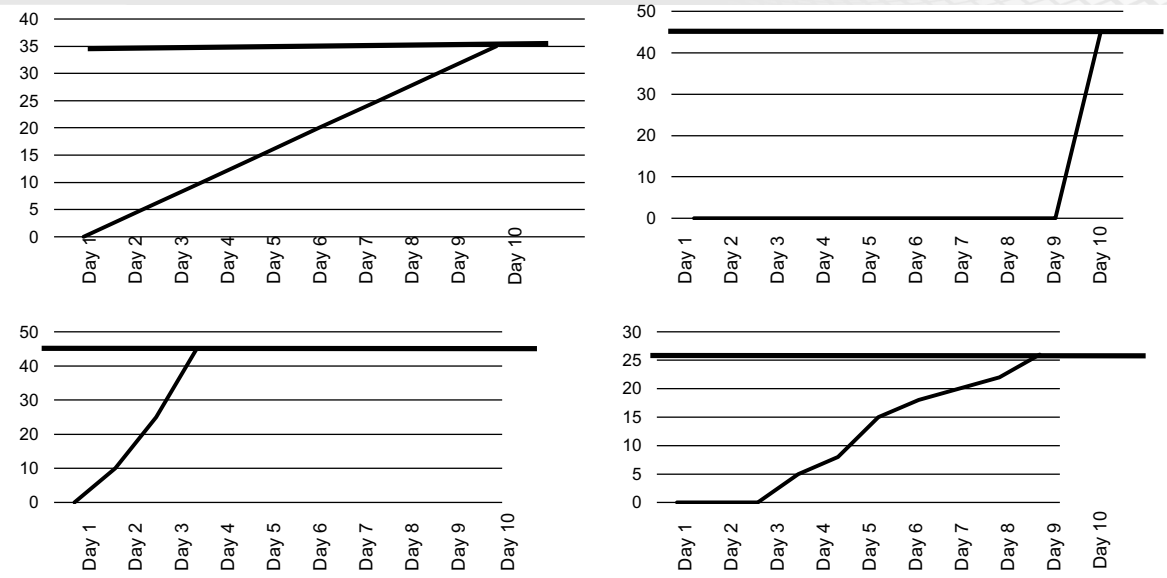


Exercise: Charts

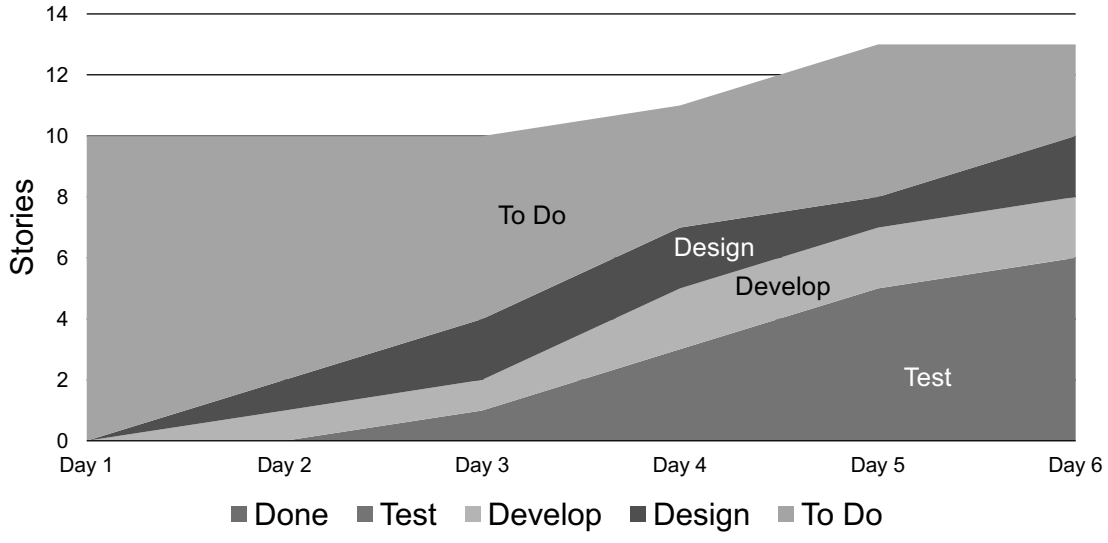
- Review the charts on the following three pages
- Discuss the status of those teams in your groups
- What problems can you see?
- How do you know?

PREPARE 5 min | SHARE 5 min

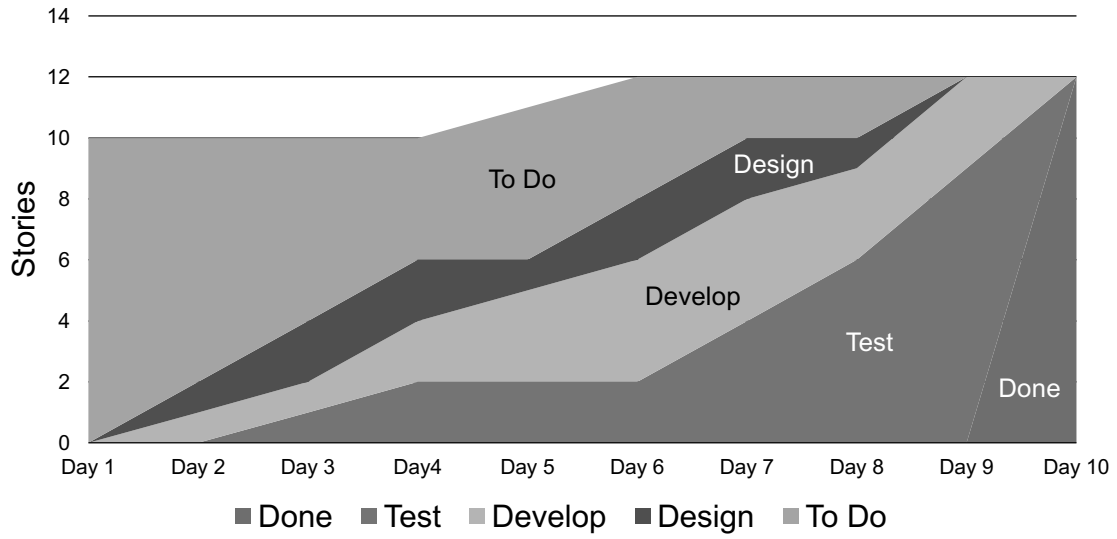
Example burn-up charts



Cumulative Flow Diagram example 1

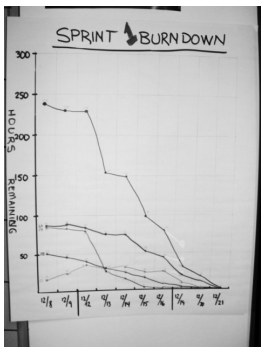


Cumulative Flow Diagram example 2

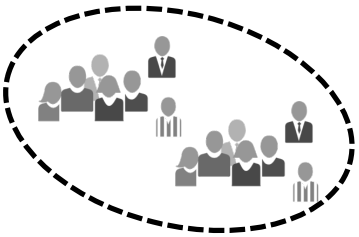


Iteration burn-down

- Many Scrum teams use Iteration burn-down charts
- Burn-downs count the remaining effort (Stories, tasks, etc.)
- As we don't advocate tasks in SAFe, we prefer burn-ups and CFDs
- Burn-down charts provide several other challenges:
  - Focus on tasks completed vs. Stories completed
  - Hard to distinguish between work added and not done



Collaboration with other teams



- The team should integrate their work often with other teams in the program (at least multiple times per Iteration)
- Work with the System Team on automated system level tests
- Join their Daily Stand-Up when important issues arise
- Join their demo or planning
- Work with the System Architect to better manage dependencies with other teams

Just go talk to that team! ;)

Actively engage with other Scrum Masters

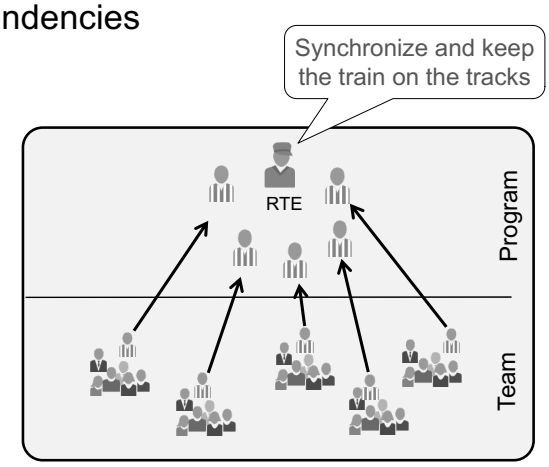
Help your team learn from others and vice versa. Self-organize with other Scrum Masters and the RTE to 'optimize the whole.'

- Actively participate in the Scrum of Scrums
- Coordinate the implementation of Program Level improvement backlog items (typically the output of the PI Inspect and Adapt session)
- Work together with other Scrum Masters to organize and maintain Communities of Practice
- Visit other teams' Scrum ceremonies and invite other teams to yours

Scrum of Scrums

Programs continuously coordinate dependencies through a Scrum of Scrums (SoS).

- The SoS is a meeting for Scrum Masters and the Release Train Engineer to gain visibility into team progress and program impediments
- It is typically held twice per week
- It is timeboxed but is followed by a 'Meet After' for problem-solving



Exercise: Mid-Iteration changes

The Product Owner comes to the team in the middle of the Iteration and wants to change the priorities of the Stories.

- Discuss the following questions in your group:
  - What should you, as the Scrum Master, do?
  - Can you think of situations where you'll react differently?
  - What about defects in production?
  - How should they be handled?

The Scrum Master's role in tracking Iteration progress

- Facilitate mid-PI re-planning.
- Encourage the team to point out as early as possible if they think they will miss Iteration goals or PI Objectives. Communicate to and from the Scrum of Scrums.
- Encourage the use of engineering practices.
- Make sure defects are not pushed to the IP Iteration.
- Facilitate preparation for the next PI.
- Support release activities.

Common anti-patterns

- Team gets no input from Scrum of Scrums
- Teams are unwilling to change or add objectives mid-PI
- Scrum Master does all the synchronization, so team is incapable of doing it themselves

4.4 Refine the backlog

The backlog refinement event

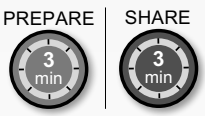
- Timebox: 1 – 2 hours weekly.
- Helps the team 'sleep' on new Stories prior to Iteration Planning.
- Provides time to identify dependencies and issues that could impact the next Iteration. Ensures that we have a ready backlog for Iteration planning.
- Agile Team members are in attendance and actively engaged, subject matter experts and other teams' members are invited as needed.

Sample Backlog Refinement Event Agenda

1. The PO presents the set of candidate Stories for the next Iteration
2. The team discusses whether the set of candidate Stories should be reduced or increased; Stories are added or removed
3. The PO guides the team through the candidate Stories one by one:
   a) The team discusses each Story, estimates it, and splits it if necessary
   b) The PO clarifies or supplements the acceptance criteria
   c) The team identifies dependencies on other teams
4. Action items are summarized for all Stories that still require external input or action

Exercise: Backlog refinement

- Discuss in your group the split in responsibility for backlog refinement between the Product Owner and the Scrum Master
- How can you, as the Scrum Master, help facilitate this process?
- A common problem is Dev Teams spending too much time refining Stories. How would you facilitate this problem?
- What should the Scrum Master do if a Story does not have acceptance criteria?

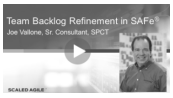


White elephant sizing

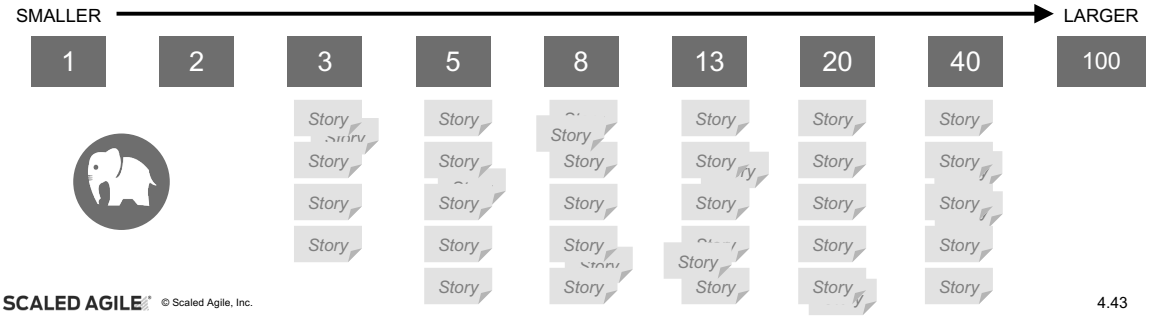
When you need to estimate a lot of Stories quickly:

- Team members take turns putting Stories under the size that they think is appropriate or using their turn to move a Story to a different estimate
- When all Stories are estimated, the team reviews the Story sizes and can make one final change

Learn more



Watch the video on the SAFe Community Platform



The Scrum Master's role in backlog refinement

- Maintain timeboxes
- Maintain the right level of a deep backlog vs. ready backlog for two Iterations
- Make sure all the team members participate
- Invite the right subject matter experts
- Hold the event at regular intervals

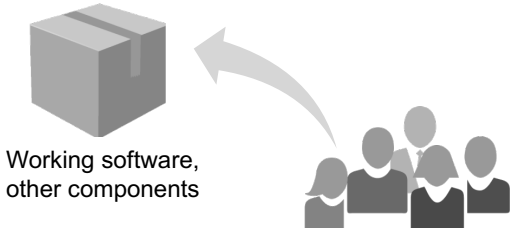
Common anti-patterns

- Arriving to the Iteration with non-ready Stories
- Not doing the backlog refinement consistently
- Team sees Stories for the first time during Iteration or PI Planning
- Feature estimations impact Story estimation

4.5 Facilitate the Iteration Review and System Demo

The Iteration Review

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



Note: This is the demo *in the Iteration*, not the *System Demo*.

Iteration Review guidelines

- Timebox: 1 - 2 hours.
- Story demonstration preparation should be limited to 1 - 2 hours. Minimize PowerPoint. Work from the repository of Stories and Enablers.
- 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	

Sample Team Iteration Review agenda

1. Review business context, Iteration Goals, and team PI Objectives
2. Demo and feedback of each Story, Enabler, and NFR
3. Discussion of Stories not completed and why
4. Current risks and impediments
5. Revised Team Backlog and team PI Objectives

Tooling is often used to facilitate the demo.

Two views from the Iteration Review

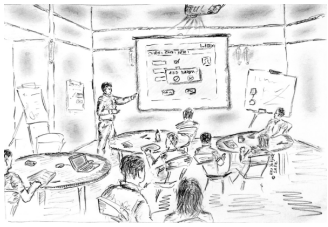
The Iteration Review provides two views into the program, based on a working system: How did we do in the Iteration? How are we doing in the PI?

How we did in the Iteration:

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

How we're doing in the PI:

- Review of PI Objectives
- Review of remaining PI scope and reprioritizing if necessary



System Demo

The System Team/Product Management /Product Owners demonstrate the Solution increment to the ART stakeholders.

- A demonstration of the integrated system to Business Owners and other program stakeholders
- Happens after the Iteration Review (may lag by as much as one Iteration, maximum!)

What does ‘done’ mean?

Our goal for the end of every Iteration:

- Working system
- No remaining work
- Everyone on same page regarding what was completed
- Standards followed
- Work accepted by the Product Owner

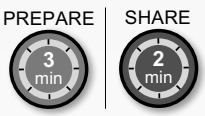
SAFe Definition of Done

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, met 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, met 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: Definition of Done

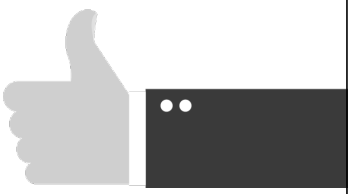
Discuss in your group:

- What is the importance of the Definition of Done?
- Should it be the same for all teams?
- What is your role, as Scrum Master, in creating and updating it?



Benefits of a Definition of Done

- Aligns expectations among team members, Product Owner, and stakeholders regarding what 'done' means
- Helps to optimize quality
- Helps maintain predictability
- Avoid unfinished work that leads to technical debt



Built-In Quality

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

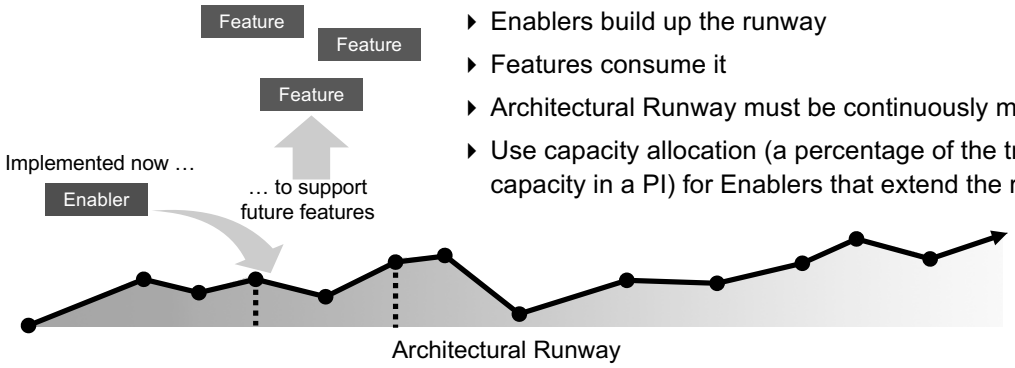
Building quality in:

- 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

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.

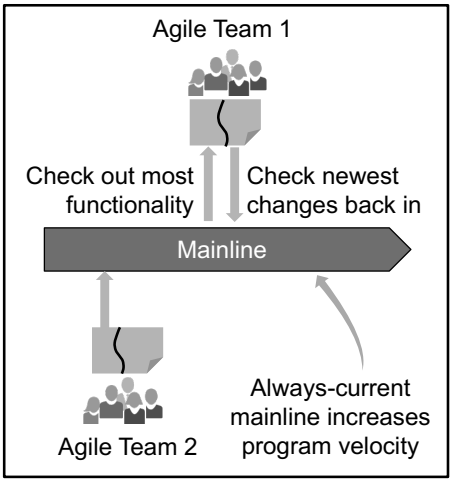


- Enablers build up the runway
- Features consume it
- Architectural Runway must be continuously maintained
- Use capacity allocation (a percentage of the train's overall capacity in a PI) for Enablers that extend the runway

Continuous system integration

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

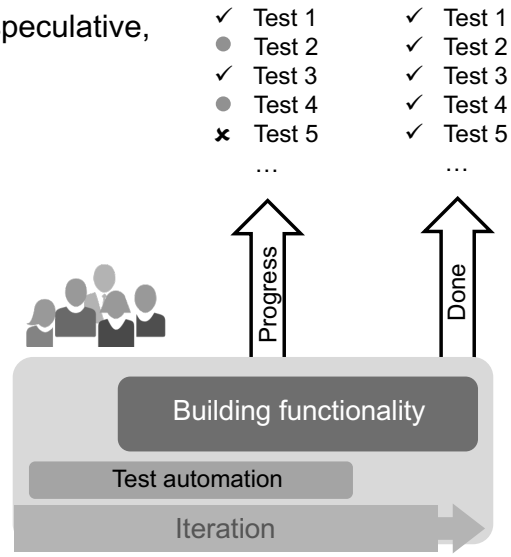
- Integrate every vertical slice of a User Story
- Avoid physical branching for software
- Frequently integrate hardware branches
- Use development by intention in case of inter-team dependencies
  - Define interfaces and integrate first, then add functionality



Test first: Automate now!

Otherwise, velocity is bottlenecked, quality is speculative, and 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 indicators



The Scrum Master's role in the Team and System Demos

- Begin to consider how and what to demo in Iteration Planning
- Make sure the right participants are present
- Ensure that the team celebrates its accomplishments and that stakeholders acknowledge them
- Make sure different team members have the opportunity to demo
- Ensure that the team is ready for the System Demo and coordinates with the System Team

Common anti-patterns

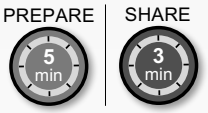
- A lot of time is spent preparing for the demo
- Demo is mainly talk/slides as opposed to working software and/or hardware
- Meeting doesn't happen if PO is unavailable
- PO sees things for the first time in the Team Demo
- System Demo is not done because 'the Team Demo is enough'
- Team members are not invited to the System Demo, to save time
- Demos that are not interesting/relevant to Program Level stakeholders

Exercise: Scrum Master's role in building quality in

Using the following four focus points, capture your team's ideas that explain why it's necessary to build-in quality from the start:

1. Higher customer satisfaction
2. Improved velocity and delivery predictability
3. Better system performance
4. Improved ability to innovate, scale, and meet compliance requirements

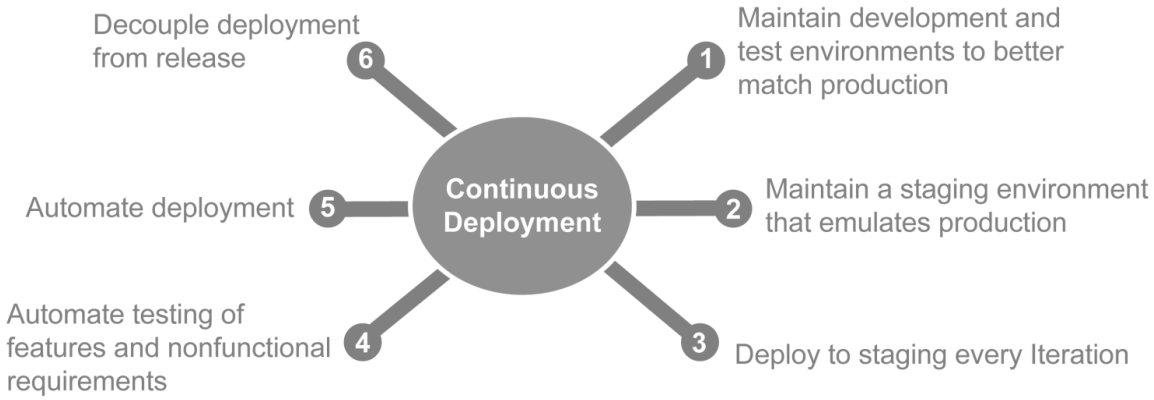
- Be prepared to share



4.6 Continuously deploy with DevOps

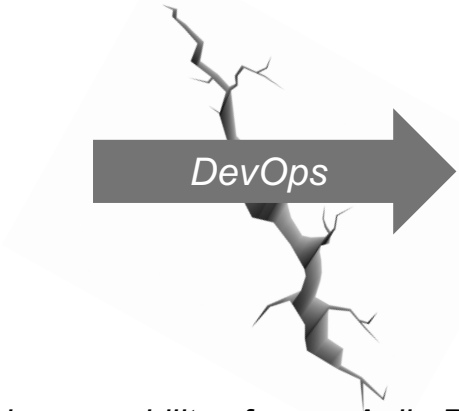
Six Recommended Practices for Continuous Deployment (CD)

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



What is DevOps?

An Agile approach to bridge the gap between development and operations to deliver value *faster and more reliably.*



Development:

- Create change
- Add or modify Features

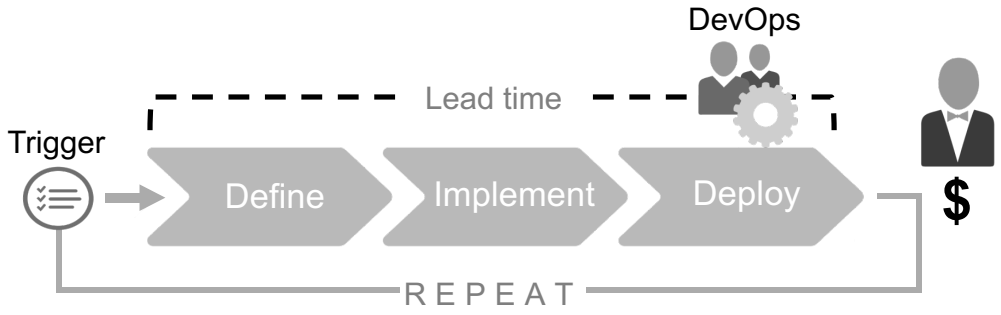
Operations:

- Create stability
- Create or enhance services

DevOps is a capability of every Agile Release Train

DevOps is IN the Value Stream

Value occurs only when the end users are operating the Solution.



DevOps isn't optional. The only question is how efficient it is.

A CALMR approach to DevOps

- **C**ulture Establish a culture of shared responsibility for development, deployment, and operations.
- **A**utomation Automate the continuous delivery pipeline.
- **L**ean flow Keep batch sizes small, limit WIP, and provide extreme visibility.
- **M**easurement Measure the flow through the pipeline. Implement application telemetry.
- **R**ecovery Architect and enable low risk releases. Establish fast recovery, fast reversion, and fast fix-forward.

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

Exercise: A CALMR approach?

How does the SAFe CALMR approach helps DevOps run more smoothly?

1. Development Team:
   - Increasing the frequency and quality of deployments
   - Improving innovation and risk-taking by making it safer to experiment
2. Scrum Master:
   - Realizing faster time to market
   - Improving Solution quality and shortens the lead time for fixes
3. IT Operations:
   - Reducing the severity and frequency of release failures
   - Improving the Mean Time to Recovery (MTTR)

PREPARE 10 min | SHARE 5 min

For Reference

DevOps is a **C**ultural shift, first of all

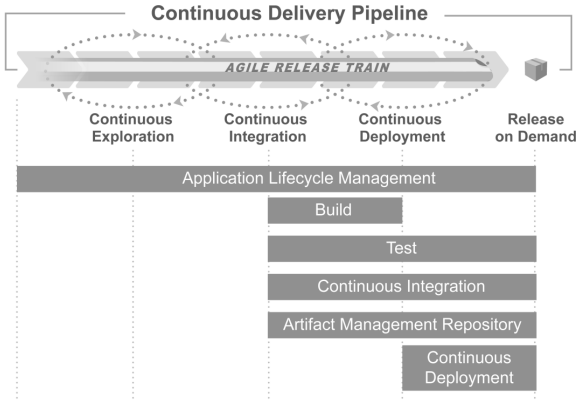
- A culture of shared responsibility for development and deployment
- DevOps requires a tolerance for failure and rapid recovery, and rewards risk taking
- Sharing discoveries, practices, tools, and learning across silos is encouraged

Automate the deployment process

For Reference

Automation of Continuous Delivery Pipeline

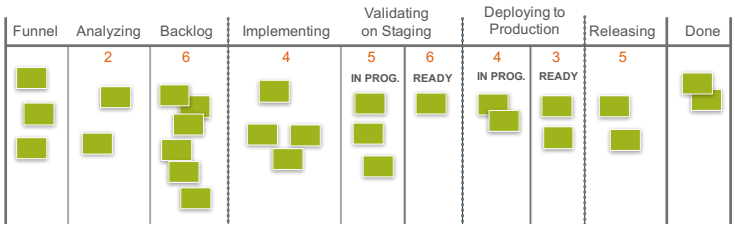
1. Match development environments to production to the extent feasible
2. Maintain a staging environment that emulates production
3. Deploy a working system to staging every Iteration
4. Automate testing of Features and performance tests
5. Automate deployment
   - Everything under version control
   - Automatically build environments
   - Automate the actual deployment process

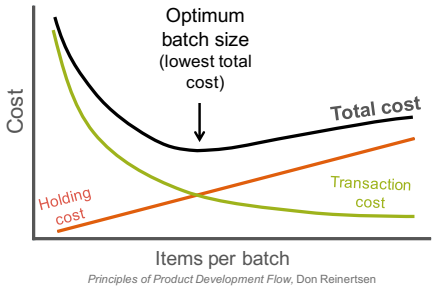


Focus on continuous Lean flow of value

For Reference

Lean Flow accelerates delivery





Principles of Product Development Flow, Don Reinertsen

- Identify bottlenecks and balance the amount of WIP against the available development and operations
- Decrease the batch sizes of the work
- Manage and reduce queue lengths

For Reference

Measure everything

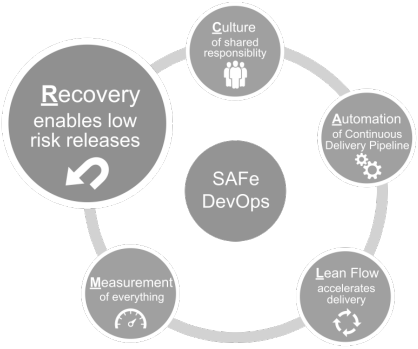
- Collect data on business, application, infrastructure, and client layers
- Collect data about the deployment pipeline itself
- Store logs in ways that enable analysis
- Different telemetry for different stakeholders
- Broadcast measurements
- Overlay measurements with events (deploys, releases)
- Continuously improve telemetry during and after problem-solving

For Reference

Architect for release-ability and **R**ecovery

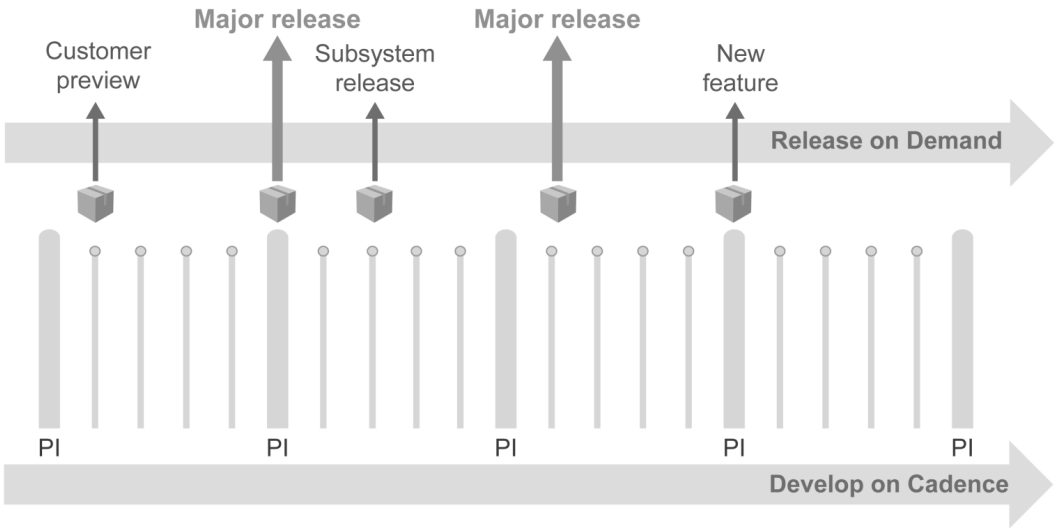
- Stop-the-line mentality
- Plan for and rehearse failures
- Build the environment for both roll-back and fix-forward
- Use tools such as:
  - Feature toggles
  - Dark launches
  - Chaos monkey
  - Canary Releases

Source: the DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations IT Revolution Press..Kim, Gene; Humble, Jez; Debois, Patrick; Willis, John

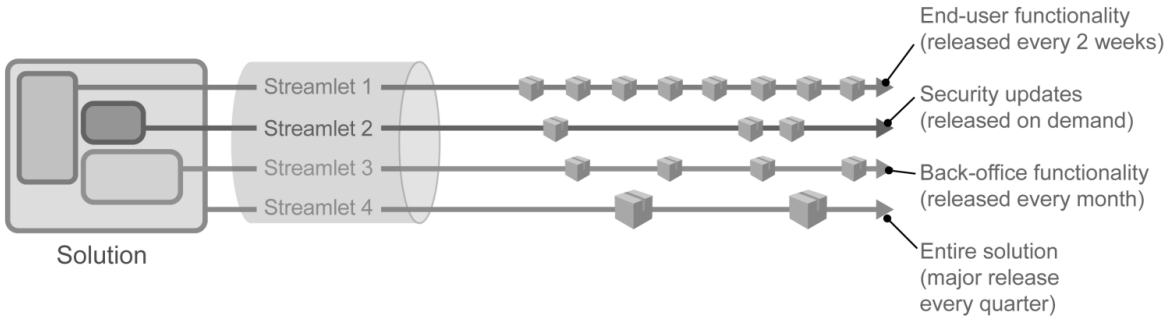


4.7 Release on Demand

Decouple deployment from release



Decouple release elements from the total solution



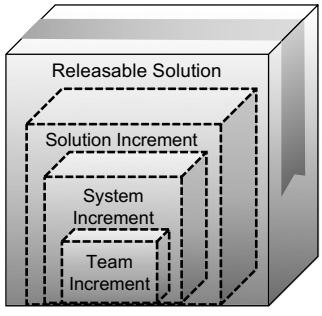
Releasing includes additional activities

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. …



Exercise: Continuous Delivery Pipeline

Implementing the Continuous Delivery Pipeline is not easy. Identify three minimum viable improvements you could implement at your organization as a Scrum Master.

1. _____

2. _____

3. _____

4.8 Facilitate relentless improvement

Relentless improvement

Agile Teams continuously adapt to new circumstances and improve the methods of value delivery.

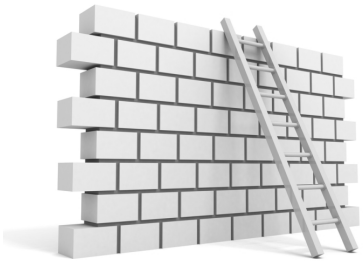
- Understand where you are
- Foster the culture of 'improving everywhere'
- Use retrospectives as summary points but not as limitations
- Support continuous learning
- Actively engage with other SMs to drive improvement on the Program Level

Improving everywhere

Address every area that surfaces as a constraint to the team's performance.

Examples:

- Testing: moving from manual to automated
- Communication with remote teams, subject matter experts, etc.
- The team's skill set
- Preparing and running the demo
- Nonfunctional Requirements (NFR) testing
- More efficient and disciplined design sessions
- Finding a better fast-food restaurant ☺
- And so on …



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
- Just the Agile Team
- Pick 1 – 2 things that can be done better or preserved, 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

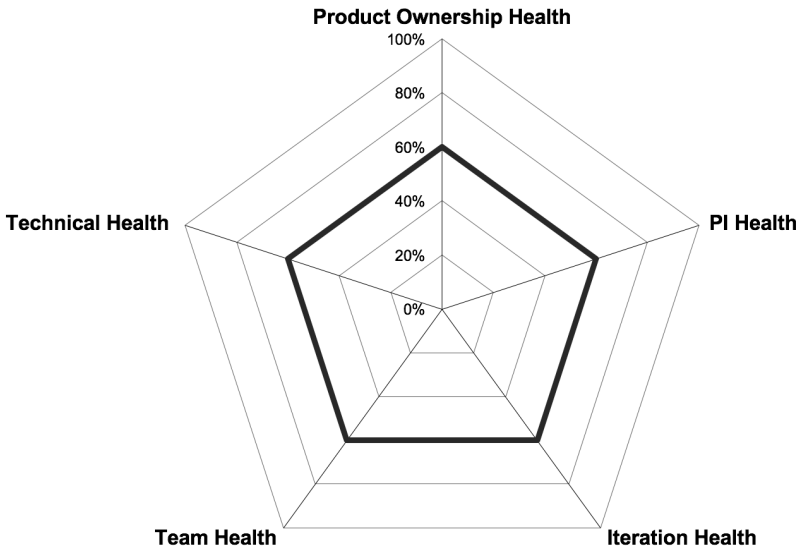
1. What went well?
2. What didn't?
3. What we can do better next time? What can we preserve?

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

Team self-assessment



ScrumMasterChecklist.org

- An example checklist for Scrum Masters
- www.ScrumMasterchecklist.org
- Focuses on 4 areas:
  1. Product Owner
  2. Team
  3. Engineering practices
  4. Organization (won't need this for SAFe)

Creative Iteration retrospectives

Simple
Three columns and open discussion

Appreciations
Has someone helped you or helped the team?

One word
to describe the Iteration

Individually
write Post-Its and then find patterns as a group

Rate the Iteration on a scale of 1 – 5 and then brainstorm how to make the next a 5

Exercise: Tune & Adjust!

- Team Level Retrospective discussion:
  - What's working well
  - What isn't working
  - What the team can do better next time
- Your retrospective should yield both quantitative and qualitative insights
- How might we apply this to the Program Level?

PREPARE 8 min | SHARE 5 min

The Scrum Master's role in the improvement

- Encourage improvement between retrospectives
- Coach the team on problem-solving techniques
- Retrospective
  - Start by reviewing the results of the previous retrospective
  - Make sure each person speaks
  - Make sure the meeting ends with actionable improvement Stories that are added to the backlog
  - Write down what people are saying exactly
  - Take program concerns to the RTE

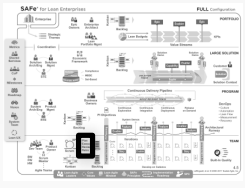
Common anti-patterns

- The only focus is on what to improve and not what to preserve
- Focus on problems that are outside of the team's control
- Failure to achieve results
- Inviting people outside the team (especially management) to the retrospective

Lesson summary

In this lesson, you:

- Learned how to facilitate the core team meetings
- Explored how to measure the team's capabilities and agility
- Understood how to help the team improve



Suggested Scaled Agile Framework reading:

- *"Iteration Execution" article*
- *"Iteration Review" article*
- *"Iteration Retrospective" article*