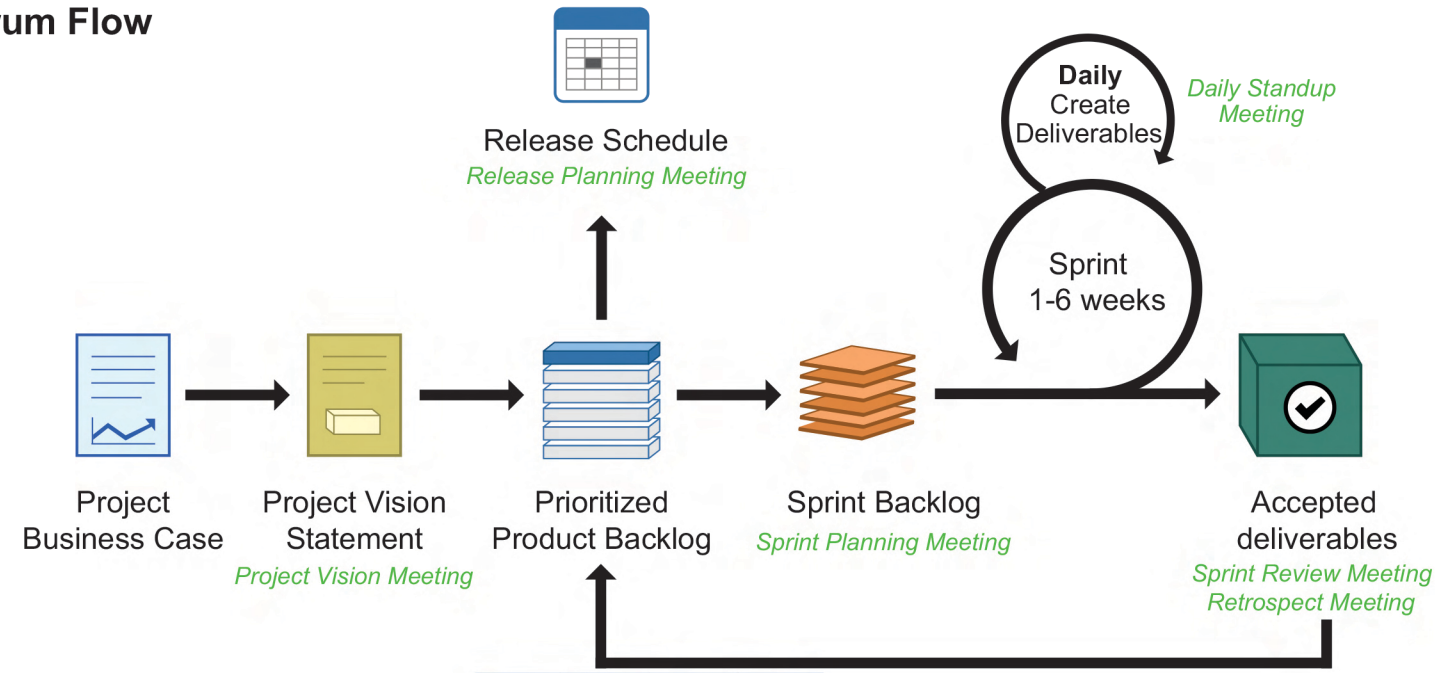
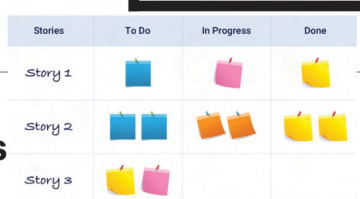
Scrum Method and how / why this needs to be firmly in place (User Story upwards) for the Scaled Agile Framework (SAFe) to work successfully at Capability Maturity Level 2+

Scrum Flow





Scrum Phases and Processes

Project Planning (PP) (PMC) Project Monitoring & Control

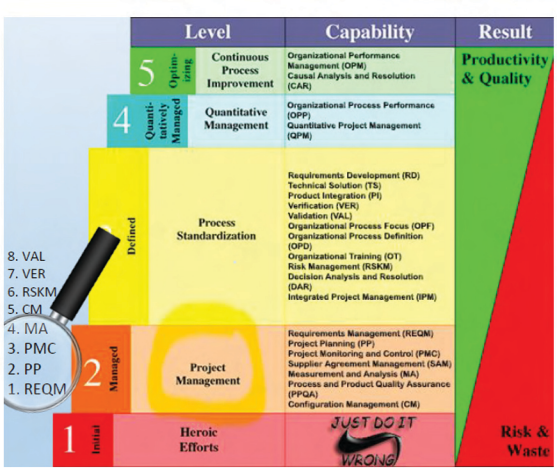
To Do In Progress Done

Initiate	Plan & Estimate	Implement	Review & Retrospect	Release
Create Project Vision	Create User Stories	Create Deliverables	Convene Scrum of Scrums	Ship Deliverables
Identify Scrum Master & Stakeholder(s)	Approve, Estimate and Commit User Stories	Conduct Daily Standup	Demonstrate and Validate Sprint	Retrospect Project
Form Scrum Team	Create Tasks	Groom Prioritized product Backlog	Retrospect Sprint	
Develop Epic(s)	Estimate Tasks			
Create Prioritized Product Backlog	Create Sprint Backlog			
Conduct Release Planning				

Development Team (Demo)
Product Owner (VAL)

Development Team

Product Owner (VER)



SCRUM ON A PAGE

SCRUM PRINCIPLES

Empirical Process Control

Scrum prescribes making decisions based on observation and experimentation rather than detailed upfront planning.

Self-organization

Scrum believes that today's workers have much more knowledge to offer than just their technical expertise and that they deliver greater value when self-organized.

Collaboration

In Scrum, product development is a shared value-creation process that needs all the stakeholders working and interacting together to deliver the greatest value.

Value-based Prioritization

Delivering the greatest value in the shortest amount of time requires prioritization and selection of what could be done from what should be done.

Time-boxing

Time is treated as a limiting constraint and time-boxing is used as the rhythm to which all stakeholders work and contribute.

Iterative Development

The customer may not always be able to define very concrete requirements. The iterative model is more flexible in accommodating changing requirements.

ROLES

CORE:

Product Owner

- Defines the Project Vision and Release Schedule as the "Voice of the Customer"
- Defines customer requirements in the form of Epics/User Stories and clarifies these requirements for team members
- Prioritizes items on the Prioritized Product Backlog according to business value
- Provides Acceptance/Done Criteria and inspects deliverable(s) to validate them

Scrum Master

- Ensures that Scrum processes are correctly followed by all Scrum Core Team members, including the Product Owner
- Ensures that an ideal project environment exists for the Scrum Team to successfully complete Sprints
- Oversees Release Planning Sessions and convenes other meetings
- Acts as a servant-leader who helps motivate and coach the team

Scrum Team

- Typically a small team of 6–10 members with no further subdivision of teams
- Cross-functional and self-organizing, the Scrum Team enjoys complete autonomy during a Sprint
- Members are generalists across domains and specialists in at least one area
- Responsibility for the work lies with the whole team

NON-CORE:

Stakeholders

- Customers
- Users
- Sponsors

Vendors

Scrum Guidance Body

ARTIFACTS

Project Vision Statement

The purpose of this artifact is to explain business need the project is intended to meet and should focus on the problem rather than the solution.

Prioritized Product Backlog

The Product Owner generates a prioritized list of requirements that, when turned into potentially shippable product functionality, will deliver the Project Vision. Owned by the Product Owner.

Sprint Goal

Proposed by the Product Owner and accepted by the team, it is a one-sentence aim for the current Sprint.

Sprint Backlog

The Scrum Team commits to executing this list of items in the upcoming Sprint. Any risk-mitigating activities are also included as tasks in the Sprint Backlog.

Impediment Log

Impediments or obstacles encountered by the team should be formally recorded by the Scrum Master in an Impediment Log.

Product Increment

The potentially shippable deliverable of the team at the end of each Sprint that satisfies the Acceptance and Done Criteria.

MEETINGS

Project Vision Meeting

This meeting explains the business need the project is intended to meet and should focus on the problem rather than the solution.

Release Planning Meeting

The purpose of this meeting is to develop a Release Plan that defines when various sets of usable functionality or products will be delivered to the customer.

Sprint Planning Meeting

The primary output of this meeting is the Sprint Backlog. Task Planning and Task Estimation are accomplished during Sprint Planning. Time-boxed to 8 hours for a 1-month Sprint.

Daily Standup Meeting

Scrum Team members gather for short, daily meeting time-boxed to 15 minutes. Each Scrum Team member answers the following three questions:

- What did I complete yesterday?
- What will I complete today?
- What impediments or obstacles am I currently facing?

Sprint Review Meeting

The Scrum Team presents the completed Sprint deliverables to the Product Owner who either accepts or rejects them based on the defined Acceptance and Done Criteria. Time-boxed to 4 hours for a 1-month Sprint.

Retrospect Sprint Meeting

Team members discuss what went well during the previous Sprint and what did not go well, the goal being to learn and make improvements in the Sprints to follow. Time-boxed to 4 hours for a 1-month Sprint.

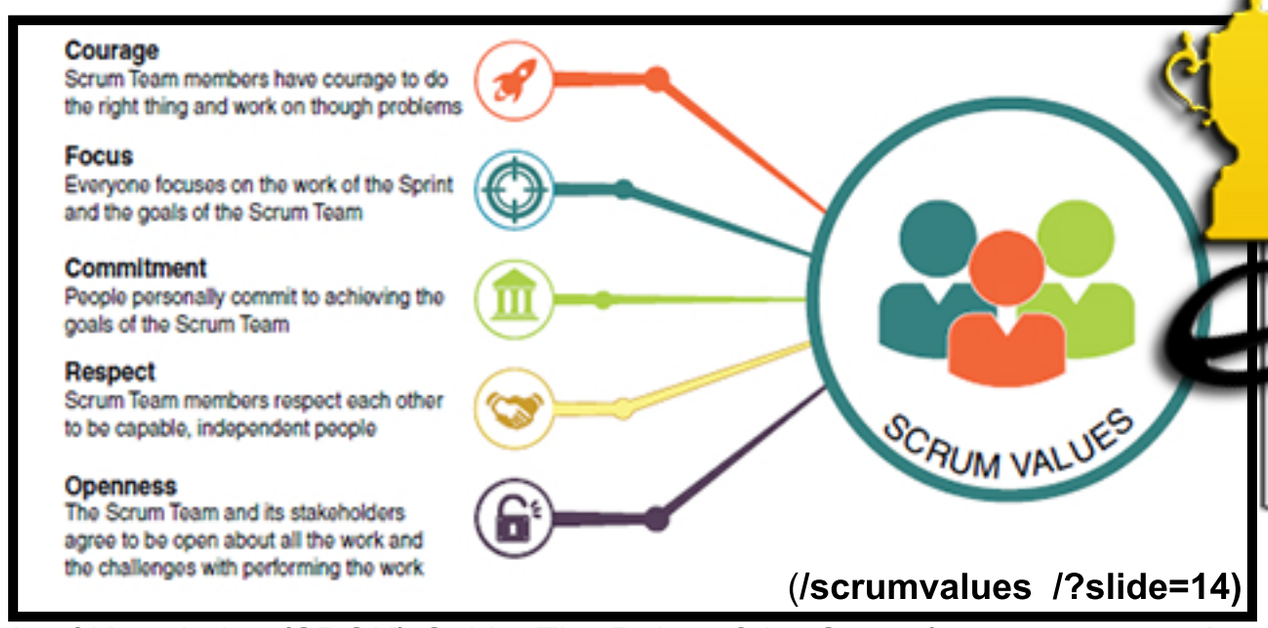
SCRUM METHOD VALUED

Focus on Respect for each other to build a winning Scrum TEAM
Do this through professional application of Scrum Principles, Processes & Aspects

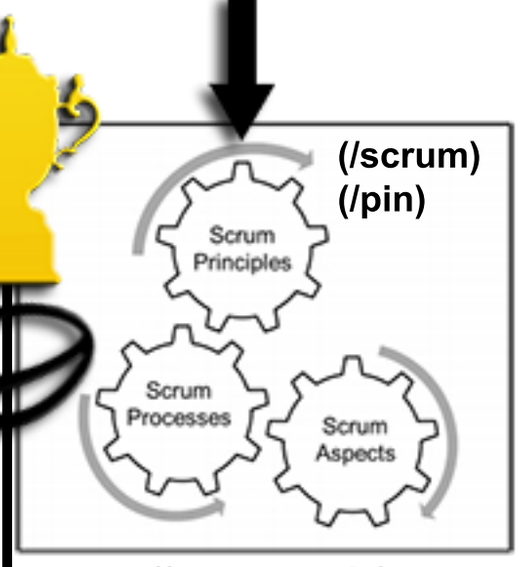
(/scrumsixmins)

Let's run Scrum @ Capability Maturity Level 2+

(/cmmi)



(/scrumvalues /?slide=14)

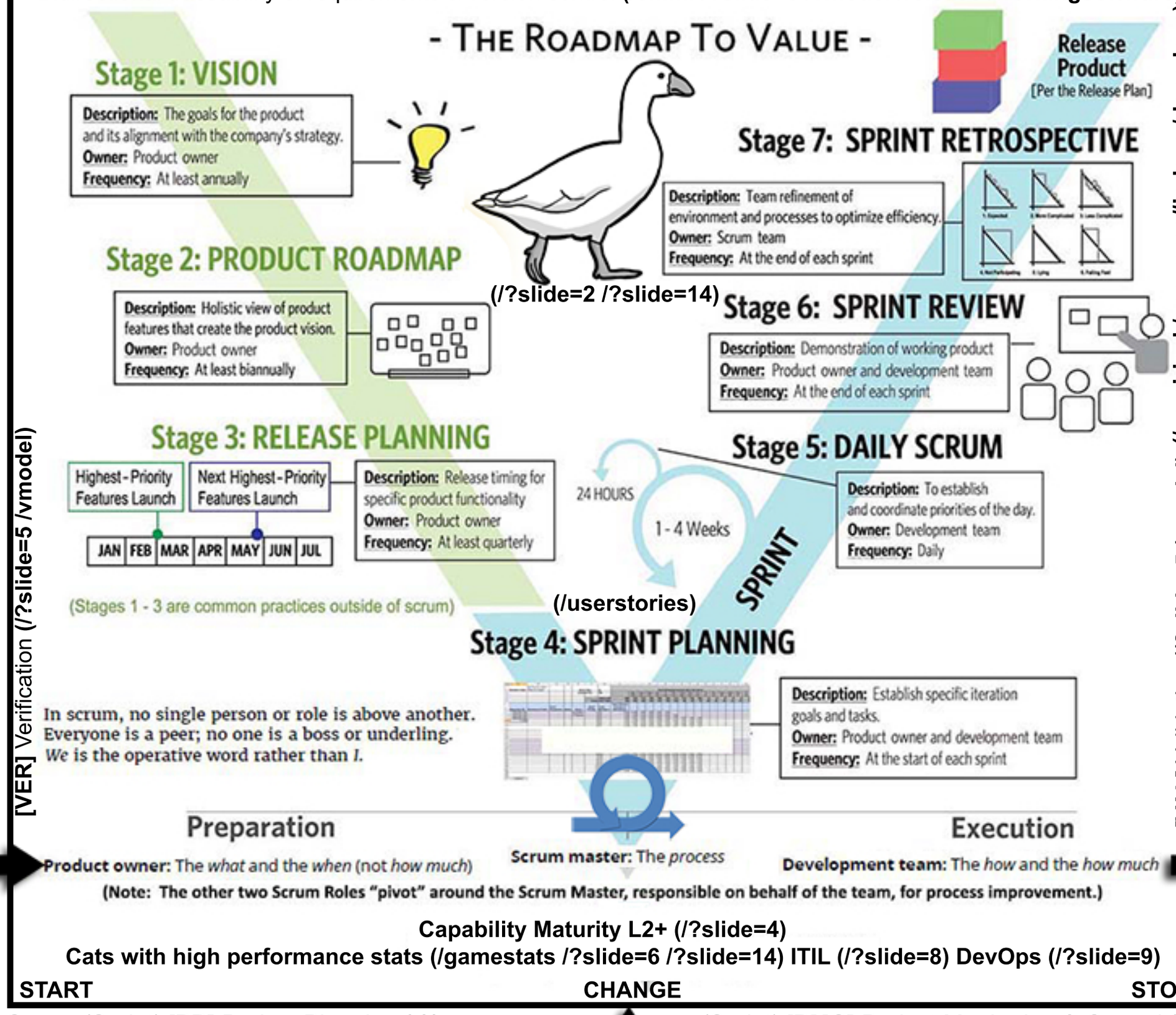


(/scrum) (/pin)

(/scrumstudy) (/studymethods)

Scrum Body of Knowledge **(SBOK)** Guide: The Rules of the Game **(www.scrumstudy.com)**

Bidirectional traceability of requirements & version control **(/cmmi /?slide=5 /vmodel /release /workingsoftware)**



(/?slide=2 /?slide=14)

(/userstories)

PRODUCT OWNER WHAT & WHEN

[VER] Verification **(/?slide=5 /vmodel)**

[VAL] Validation **(/?slide=5 /vmodel) (/empirical /scrumrca#indexc /donedone)**

HOW & HOW MUCH DEVELOPERS

At the end of each sprint does the Product Owner sign off delivery? Are the golden eggs enjoyed?

Capability Maturity L2+ (/?slide=4)

Cats with high performance stats (/gamestats /?slide=6 /?slide=14) ITIL (/?slide=8) DevOps (/?slide=9)

START **CHANGE** **STOP**

Scrum (Sprint) **[PP]** Project Planning **(/)**

(Sprint) **[PMC]** Project Monitoring & Control **(/)**

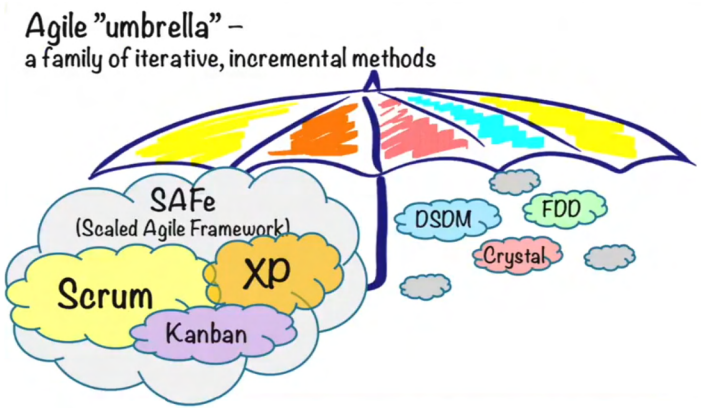
SCRUM MASTER

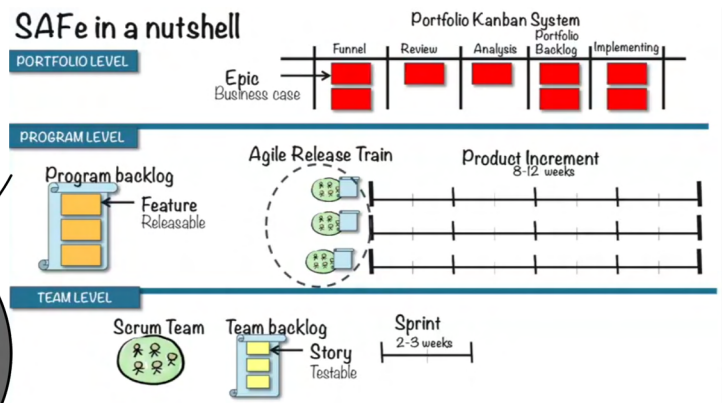
CONTINUAL (SCRUM) PROCESS IMPROVEMENT

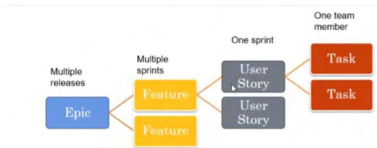
https://pmway.hopto.org (/) (not 24/7)

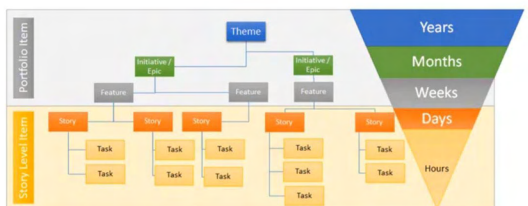
Latest version 2020/06/01 (/scrumvalued)

Scrum & SAFe

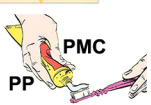






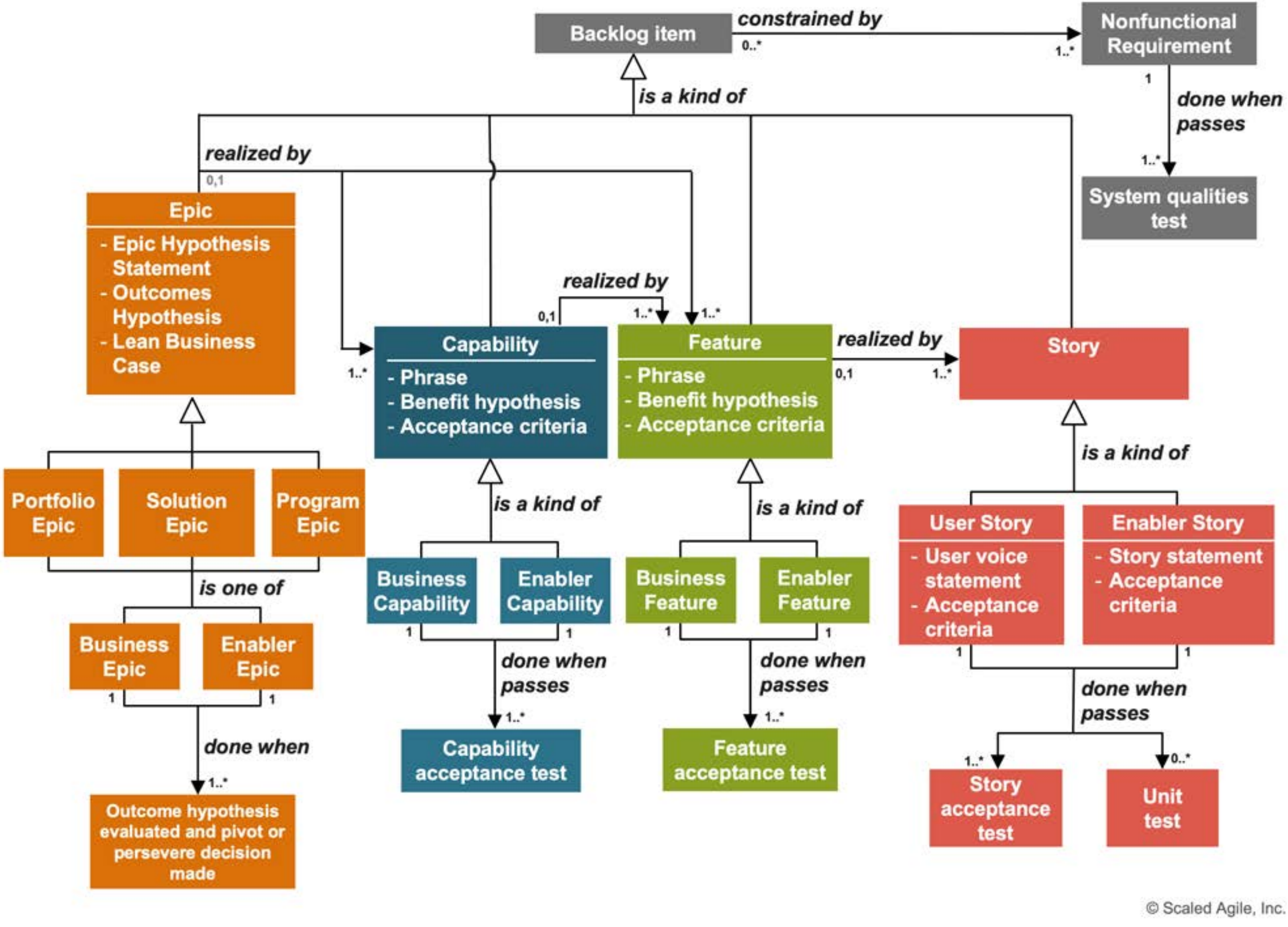


With respect to SAFe, Project (or phase or iteration) Planning (PP / User Stories) and Project Monitoring and Control (PMC / Approved Demo to / by Product Owner) at Capability Maturity Level 2+ is how Done Done Scrum teams create value.

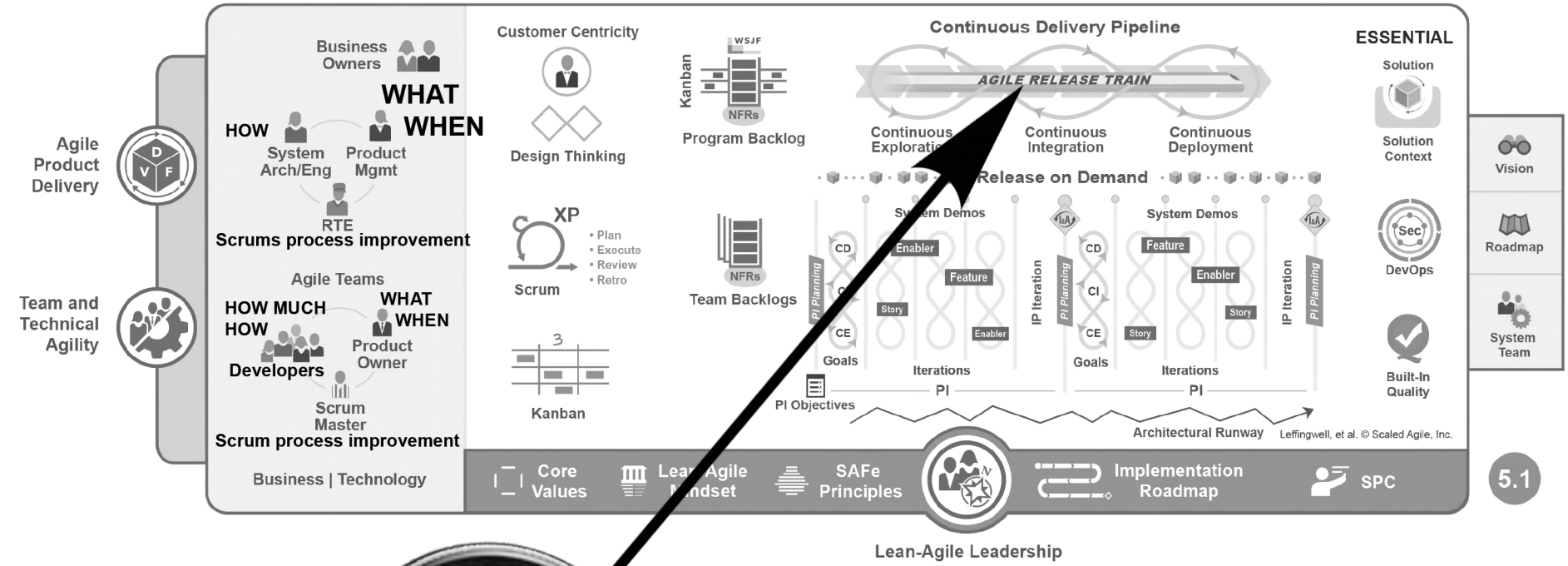


User Stories (in Features) are the building blocks of Scrum and therefore SAFe

The Product Owner approves User Stories (or Feature of finished User Stories) in Demonstrate and Validate Scrum process #16

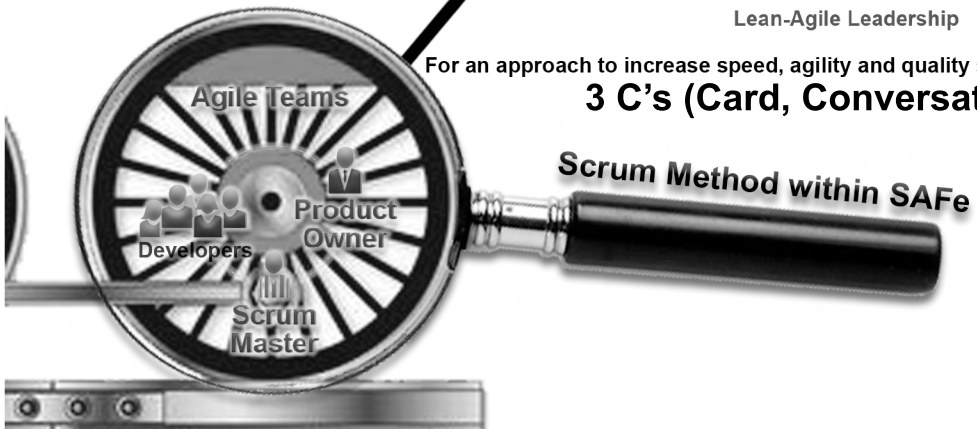


SAFe® for Lean Enterprises



For an approach to increase speed, agility and quality search for “Mike Cohn” & “User Stories” on Youtube

3 C’s (Card, Conversation and Confirmation)



19. Essential SAFe

Find section “Essential Team and Program Roles” and sub sections “What happens without...”

Simplicity—the art of maximizing the amount of work not done—is essential.

—Agile Manifesto

Overview

Throughout this book, we’ve described the Scaled Agile Framework (SAFe) in some detail. While it is indeed “distilled” from the larger body of knowledge,[1] the fact remains that it’s a significant framework. Not surprisingly, we’ve observed that not every implementation realizes the full business benefits others achieve. When diagnosing these cases, we typically discover the root causes, which illustrates what happens when some essential SAFe practice is missing.

1. www.scaledagileframework.com

It’s easy to see how that can happen. After all, it’s a big framework, designed for people building big systems. And since it’s a framework—not a method or recipe—some interpretation and context-specific implementation are typically required. So, the question arises, how would an enterprise know what’s essential?

With that in mind, this final chapter introduces the “Essential SAFe” configuration, which provides the minimum—and essential—elements necessary to receive the majority of the business benefits.

Figure 19-1 illustrates the most essential elements of SAFe. This configuration of SAFe serves two purposes: one, it provides an excellent starting point for those implementing SAFe, and two, it can serve as a diagnostic assessment for those who are further down the road. And to make things as simple as possible for the reader, this chapter briefly summarizes the most critical elements of SAFe

so that it can also serve as an introductory, stand-alone guide for either of these two purposes.

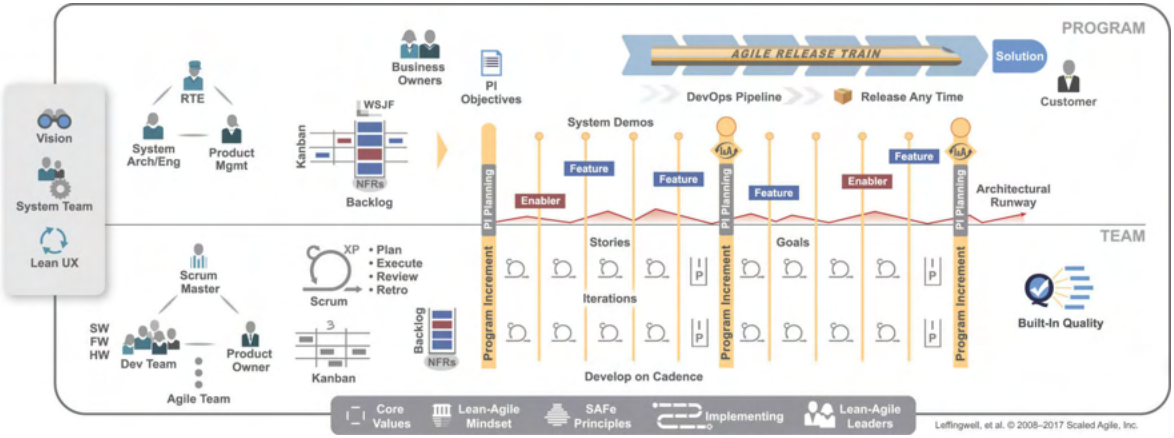


Figure 19-1. Essential SAFe is a subset of SAFe

The ten essential elements are as follows:

1. Lean-Agile principles
2. Agile teams and Agile Release Trains (ARTs)
3. Cadence and synchronization
4. Essential team and program roles
5. Program Increment (PI) planning
6. System demo
7. Inspect and Adapt (I&A)
8. Innovation and Planning (IP) iteration
9. DevOps pipeline
10. Lean-Agile leadership

Each is described in the sections that follow, along with a set of symptoms that typically occur when that element is missing from an implementation.

Lean-Agile Principles

An understanding of the nine fundamental Lean-Agile principles is critical to appreciate what SAFe is and how it helps the enterprise accomplish what it does. The following sections are abbreviated versions of the principles to help you focus on the most important aspects of the principles.

Principle #1: Take an Economic View

The over-arching goal is simple: to deliver the best value and quality to people and society in the shortest sustainable lead time. Doing so requires a fundamental understanding of the economics of the mission and makes sure that everyday decisions are made in a proper economic context. The

primary aspects include developing and communicating the strategy for incremental value delivery and the creation of the economic framework. This helps define the trade-offs between risk, Cost of Delay (CoD), and operational and development costs, while supporting decentralized decision-making.

Principle #2: Apply Systems Thinking

W. Edwards Deming, one of the world's foremost systems thinkers, constantly focused on the larger view of problems and challenges faced by people building and deploying all types of systems: manufacturing, social, management, and even government. One central conclusion was the understanding that the problems faced in the workplace were a result of a series of complex interactions that occurred within the systems the workers used to accomplish their tasks. In SAFe, systems thinking addresses this and is applied to the organization that builds the system, as well as the system under development.

Principle #3: Assume Variability; Preserve Options

Traditional design and life-cycle practices drive picking a single requirement and design option early in the development process when there is high uncertainty. However, if the starting point is wrong, then future adjustments take too long and can lead to a suboptimal long-term design. Alternatively, a better approach is to maintain multiple requirements and design options for a longer period in the development cycle. Empirical data is then used to narrow focus, resulting in a design that creates better economic outcomes.

Principle #4: Build Incrementally with Fast, Integrated Learning Cycles

Lean-Agile teams develop solutions incrementally in a series of short iterations. Each results in an integrated increment of a working system. Subsequent iterations build upon the previous ones. Increments of the solution provide the opportunity for fast customer feedback and risk mitigation and also serve as minimum viable products or prototypes for market testing and validation. In addition, these early, fast feedback points allow teams to "pivot" when necessary to an alternate course of action.

Principle #5: Base Milestones on Objective Evaluation of Working Systems

Lean-Agile teams and customers have a shared responsibility to assure that investment in new solutions will deliver economic benefit. The sequential, phase-gate development model was designed to meet this challenge. But experience has shown that it does not mitigate risk as intended. In Lean-Agile development, each integration point provides an objective milestone in which to evaluate the solution frequently and throughout the development life cycle. This objective evaluation provides the financial, technical, and fitness-for-purpose governance needed to assure that continued investment will produce commensurate return.

Principle #6: Visualize and Limit Work in Process, Reduce Batch Sizes, and

Manage Queue Lengths

Lean-Agile teams strive to achieve a state of continuous flow, allowing new capabilities to move quickly and visibly from concept to cash. The following are three keys to ensure flow:

- Visualize and limit the amount of Work in Process (WIP) to restrict demand to actual capacity.
- Reduce the batch sizes of work items to facilitate reliable flow through the system.
- Manage queue lengths to reduce the wait times for new capabilities.

Principle #7: Apply Cadence; Synchronize with Cross-Domain Planning

Cadence transforms unpredictable events into predictable ones and provides a rhythm for development. Synchronization causes multiple perspectives to be understood, resolved, and integrated at the same time. Applying development cadence and synchronization, coupled with periodic cross-domain planning, offers the Lean-Agile tools needed to operate effectively in the presence of product development uncertainty.

Principle #8: Unlock the Intrinsic Motivation of Knowledge Workers

Lean-Agile leaders understand that knowledge workers generally aren't motivated by incentive compensation approaches. Such individual Management by Objectives (MBOs) causes internal competition and destruction of the cooperation necessary to achieve the larger system aim. Providing autonomy, mission, and purpose while minimizing constraints leads to higher levels of employee engagement and results in better outcomes for customers and the enterprise.

Principle #9: Decentralize Decision-Making

Achieving fast value delivery requires fast, decentralized decision-making, as any decision escalated introduces delay. In addition, escalation can lead to lower-quality decisions because of the lack of local context and changes to facts that may occur during the wait time. Decentralized decision-making reduces delays, improves product development flow, and enables faster feedback and more innovative solutions. Some decisions, however, are strategic global choices that have economies of scale warranting centralized decision-making. Since both types of decisions occur, creating an established decision-making framework is a critical step in ensuring the fast flow of value.

What Happens If Your Implementation Is Not Based on SAFe Principles?

- Change agents, management, and other practitioners are unable to effectively adapt practices.
- Improvement of business outcomes over time is impaired; practices and measures that used to be beneficial become constraints.
- Lean-Agile mindset is unachievable; Agile practices deployed on a wrong-minded "mental platform" produce serious problems.
- Misalignment, conflict, and disagreement on processes and practices are impossible to

resolve.

Agile Teams and Release Trains

ARTs are virtual organizations that have all the people they need to define and deliver value. Each ART is a long-lived, self-organizing team of Agile teams, a virtual organization (50–125 people) that plans, commits, and executes together. ARTs are organized around the enterprise's significant value streams and exist solely to realize the promise of that value by building solutions that deliver benefit to the end user. The ART aligns teams to a common mission via a single vision and program backlog.

In functional organizations, developers work with developers, testers work with other testers, and architects and systems engineers work with each other. While there are reasons why organizations have evolved that way, value doesn't flow easily, as it must cross *all* the silos. Daily involvement of managers and project managers is necessary to move the work across the silos. As a result, progress is slow, and handoffs and delays rule the day.

Instead, the ART takes a systems view and builds a cross-functional organization that includes all people needed to continuously define, build, test, and deploy valuable features. This means each train must include cross-functional Agile teams, as well as people from various other disciplines, as shown in Figure 19-2.

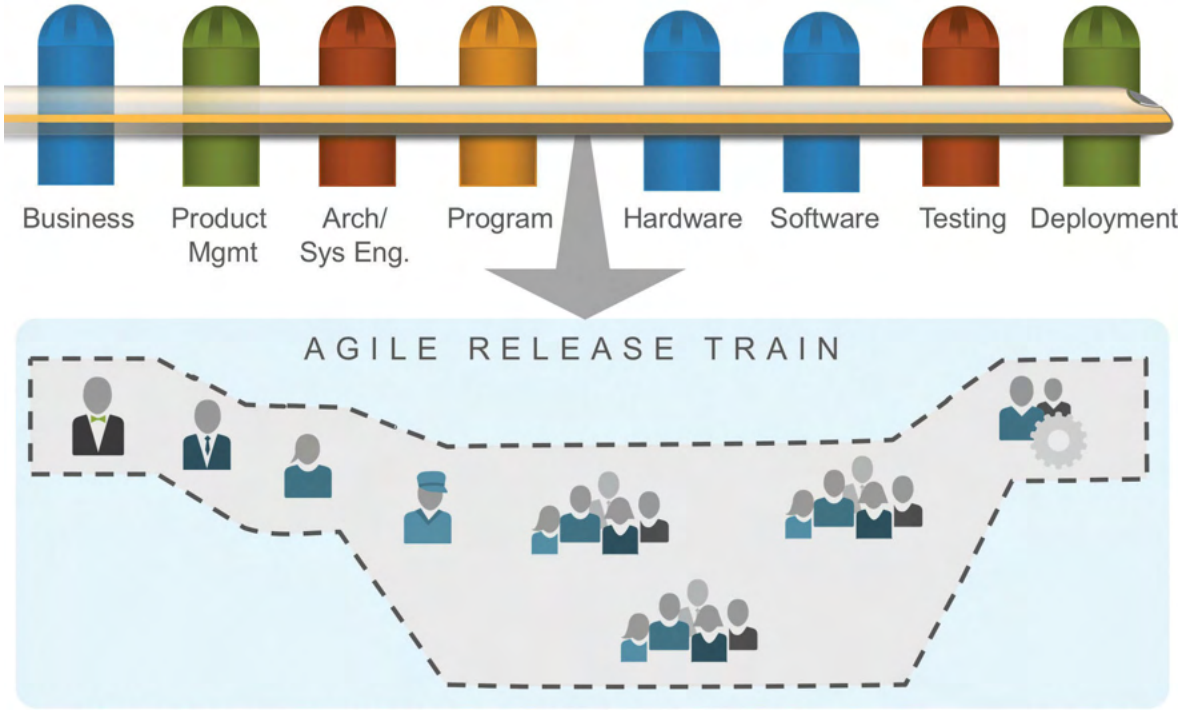


Figure 19-2. ARTs are cross-functional

In support of this, Agile teams are cross-functional too, as shown in Figure 19-3.

Figure 19-3. Agile teams are cross-functional

Each Agile team has the skills and people needed (for example, designers, developers, and testers) to effectively deliver a feature or component with a minimum number of dependencies on others.

Together, this fully cross-functional organization—whether physical (direct organizational reporting) or virtual (line of reporting is unchanged)—has everyone and everything necessary to define and deliver value. It is self-organizing and self-managing at the team and program levels. This creates a far leaner organization, one where traditional daily task management is no longer required. Value flows more quickly, with a minimum of overhead. That's the purpose of the ART.

What Happens When Agile Teams and ARTs Aren't Implemented Correctly?

- Teams are isolated and struggle to deliver value.
- There is a lack of collaboration among teams.
- Components become over-specialized, and bottlenecks are created.
- Teams are not aligned to a common mission.
- Release of value is late and problematic.
- There is no architectural and user experience integrity.
- There is too much WIP and multitasking, which leads to lower productivity and quality.

Cadence and Synchronization

Cadence provides a rhythmic pattern, the dependable heartbeat of the development process. It

makes routine that which can be routine. This enables teams to focus on leveraging the variability inherent in solution development.

Synchronization causes multiple perspectives to be understood and resolved at the same time. For example, synchronization is used to pull the disparate assets of a system together to assess solution-level viability. It's also used to align development teams and the business to a common mission at PI planning.

Together, cadence and synchronization are the primary tools used to manage the inherent variability of Research and Development (R&D).

The first cadence of SAFe is the iteration cycle; each is a Plan-Do-Check-Adjust (PDCA) learning cycle. During this short period, the team plans, builds, integrates, and demonstrates the result, followed by a short retrospective. As illustrated in Figure 19-4, iterations provide the basic cadence, or tempo, for development.

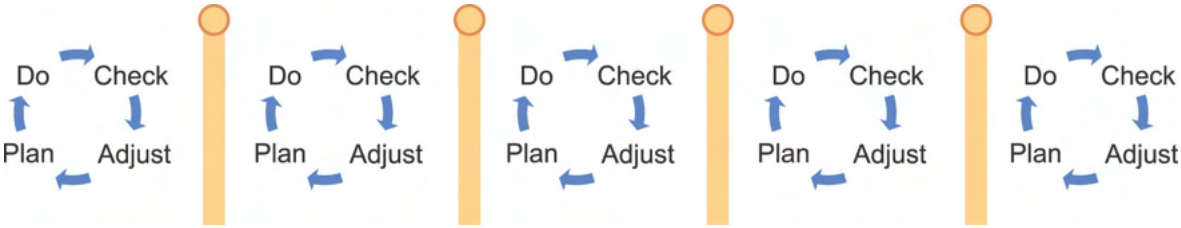


Figure 19-4. The basic iteration cadence, each a Plan-Do-Check-Adjust cycle

However, because significant solutions require integration across multiple teams, it is critical for the teams to work on a common cadence, using the same iteration duration. Otherwise, the teams may be iterating, but integration is difficult, and the slowest cycle delays problem discovery. The net result is that the teams may be iterating, but the system isn't, as shown in Figure 19-5.

To address this, SAFe provides two synchronized PDCA feedback loops. The inner team's PDCA loops are synchronized; all teams start and end the iterations at the same time. The inner loop iterations occur inside an ART PDCA cycle, or PI, as Figure 19-6 illustrates.

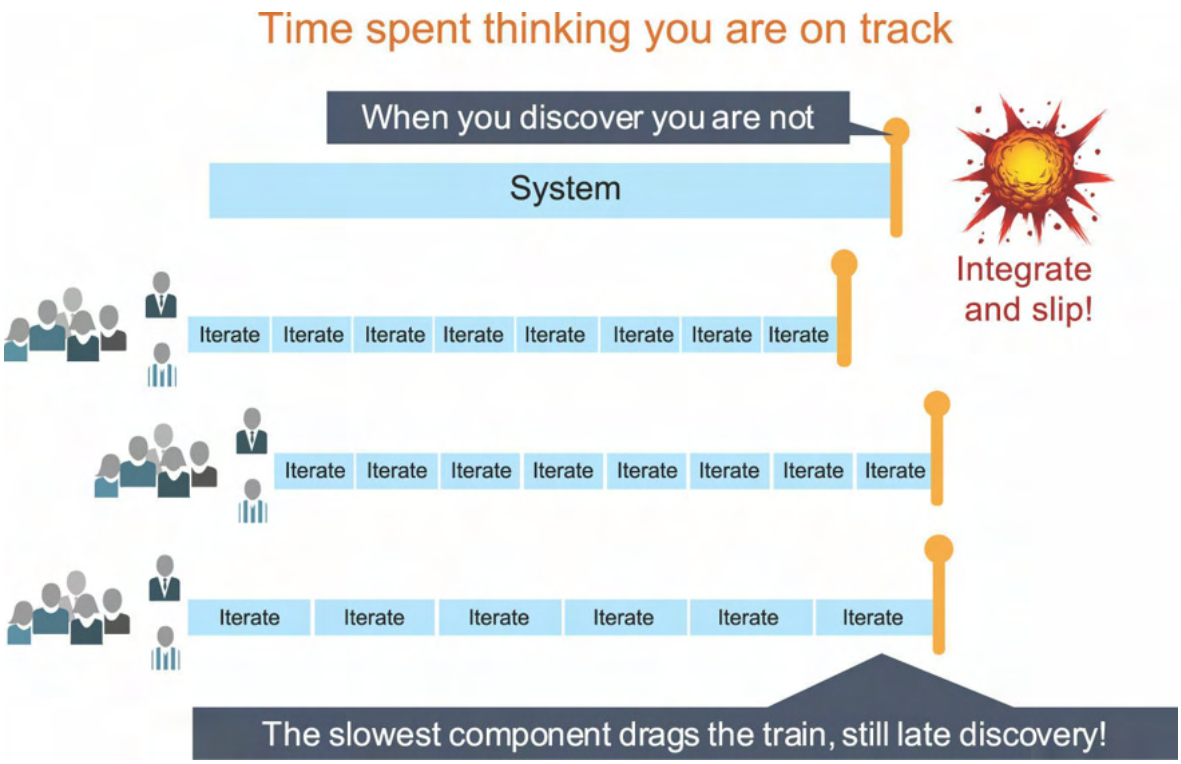


Figure 19-5. Cadence without synchronization is not enough

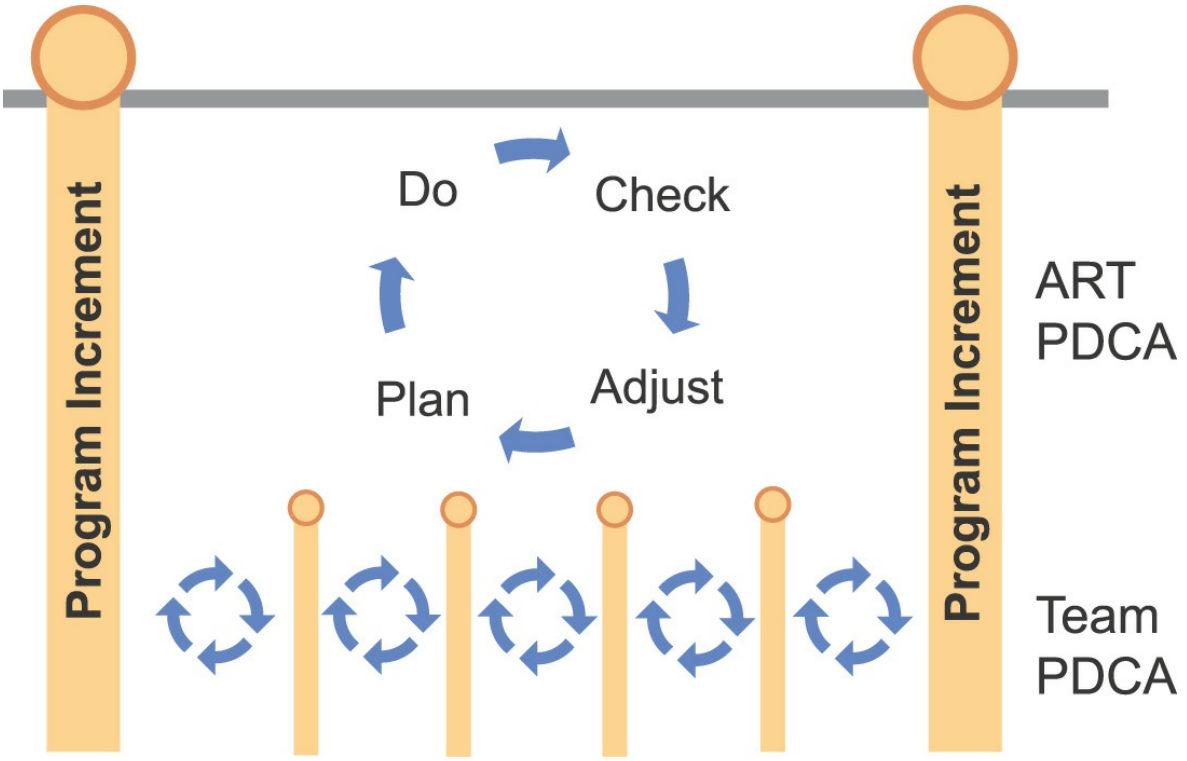


Figure 19-6. Iterations and PIs provide nested learning cycles

The PI provides a larger and more strategic PDCA timebox in which to accumulate and assess system-level performance. It also provides the cadence for the entire train to perform cross-domain planning, integration, demo, and I&A.

Develop on Cadence; Release Any Time

We've described how cadence and synchronization are the primary tools in managing the inherent variability of R&D. But it's important to note that the development cycle is distinct from the release cycle. These are two separate concerns. While in some situations the train may release solutions at the end of a PI, other programs may need to release less or more frequently than the PI cadence. Still others will have multiple, independent release cycles for the various subsystems of the solution.

What Happens without Cadence and Synchronization?

- There is gradual decline into disorder and lack of predictability.
- Getting the right people to meetings is impossible.
- Integration comes late, resulting in slips of value delivery.
- There are no forced integration and evaluation points.
- Individual teams may be Agile, but the system is not iterating.
- Enterprise agility is not achievable.

Essential Team and Program Roles

Effective implementations require the right people in the right roles, fulfilling the right set of responsibilities.

Essential Team Roles

- *Scrum Master*. Scrum Masters coach a high-performing and self-managing team. They do that by facilitating team meetings, driving Agile behavior, removing impediments, helping maintain the team's focus by protecting the team, and attending ART sync meetings.
- *Product Owner*. The Product Owner maintains the team backlog, acts as the customer for team questions, prioritizes the work, and collaborates with Product Management to plan PIs.
- *Development Team.* Developers, testers, and various specialists create and refine user stories and acceptance criteria. They define, build, test, and deliver software, hardware, and other system components or features.

Essential Program Roles

- *The Release Train Engineer (RTE)*. The RTE acts as the chief Scrum Master for the train. RTEs facilitate ART-level events and meetings and help drive Agile behavior with the teams, the ART, and other stakeholders. They help manage risk, coordinate dependencies, and coach the ART to relentless improvement.
- *Product Management.* Product Management is responsible for identifying customer needs. They own the vision and product backlog, and they sequence features for optimal Return on Investment (ROI). They drive PI objectives and release content via prioritized features and acceptance criteria.
- *System Architect-Engineering*. System Architect-Engineering aligns ARTs with a common technological and architectural vision. They participate in defining the system and subsystems and their interfaces, validating technological assumptions, and evaluating alternatives. They support robust system development by providing, communicating, and evolving the larger technological and architectural view of the solution.
- *Business Owners*. Business Owners are a small group of stakeholders who have the ultimate fiduciary, governance, and ROI responsibility for the value delivered by a specific ART. Business Owners typically have management responsibility in one or more areas, such as customer relationships, development, solution quality, deployment, operations, Product Management, and architecture. They work with the ART to continuously define the business value of plans and working systems.
- *Customer*. The customer is whoever consumes the work of an ART. Whether internal or external to the development organization, they are an integral part of the value stream and thereby participate in ART events.

What Happens without the Essential Team and Program Roles?

- Responsibilities for requirements, design, architecture, implementation, and deployment are unclear.
- Meetings are less productive and end without clear outcomes.
- Teams find it hard to integrate because of incompatible components.
- Vision and requirements are not clear, and there is a lack of prioritization.
- Deliverables do not meet stakeholder expectations.
- It is difficult to improve processes.

PI Planning

There is no other, more powerful event than PI planning. It is the cornerstone of the PI—which provides the rhythm for the ART.

It's amazing how much alignment and energy are created when there are 100 or so people all working together toward a common mission, vision, and shared purpose. Gaining that alignment in just two days can save weeks, if not months, of delays waiting on decisions and coming to agreement via a flurry of emails.

More importantly, this event represents a critical and cultural milestone for the implementation of SAFe.

- The teams come together periodically to better define and design the system that fulfills the vision and commit to near-term PI objectives.
- The ART uses this event to create, foster, and sustain a sense of shared mission, responsibility, and cooperation and collaboration.
- The responsibility for planning moves from central authority to the teams who do the work; this sends a signal of true change from management that the teams are now empowered.
- It builds the social network that the ART depends on; after all, building large-scale, complex solutions is a social endeavor.

Whenever possible, attendees include all members of the ART. After all, they are the ones doing the work, so they are the only ones who can design the system, plan, and then commit to the plan.

What Happens without Effective PI Planning?

- Stakeholders and teams don't have a clear understanding of the vision.
- Teams don't know the business context and the most important objectives.
- There is a lack of alignment between business and technology.
- Dependencies are discovered too late.
- Planning is centralized and ineffective.
- Teams are not committed to the business objectives.

System Demo

The primary measure of ART progress is the objective evidence provided by a working solution in the system demo. Every two weeks, the full system—the integrated work of all teams on the train for that iteration—is demoed to the train's stakeholders. (This is in addition to each team's iteration demo.) Stakeholders provide the feedback the train needs to stay on course and take corrective action, as shown in Figure 19-7.

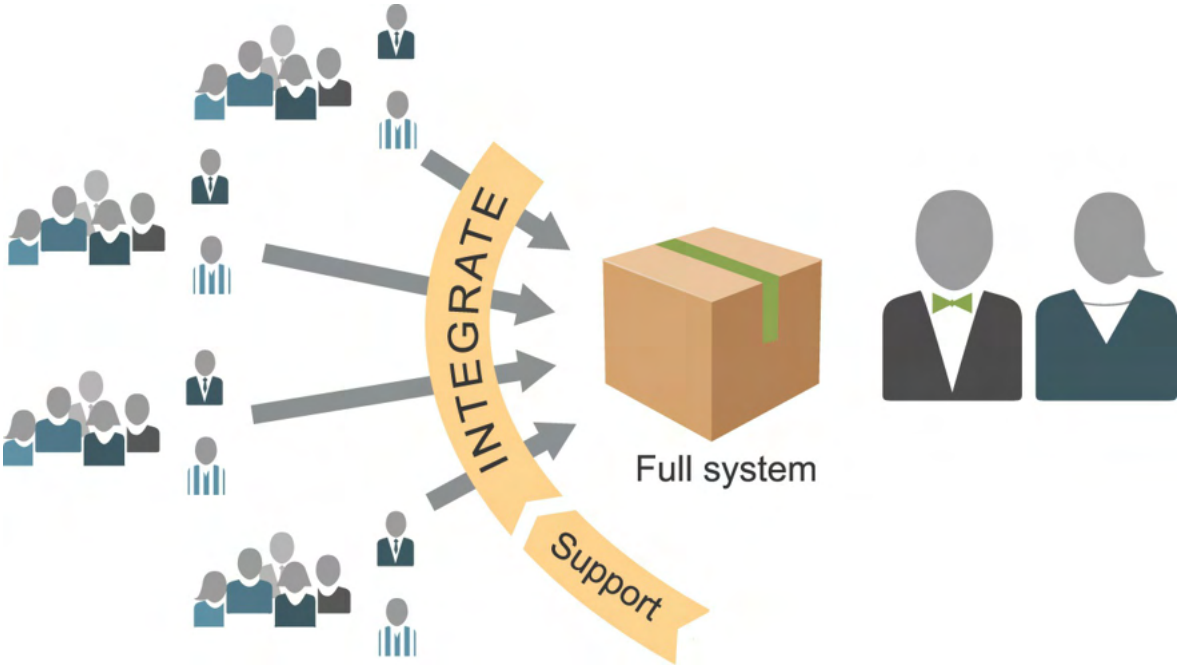


Figure 19-7. Teams integrate every iteration to demo the full system

At the end of each PI, a final system demo is held. That demo is a significant and somewhat more structured affair, as it demonstrates the accumulation of all the features (from all teams on the train) that have been developed over the course of the PI.

What Happens without the System Demo?

- There is late discovery of integration problems.
- Business Owners are unclear about solution progress, which reduces trust between the business and development organizations.
- Management relies on traditional proxy metrics.
- Quality and velocity are uncertain.
- There is little to no meaningful feedback.
- There is no forcing function for continuous integration and test automation.
- Teams might be iterating, but the system as a whole is not.

Inspect and Adapt

The I&A workshop is a significant event held at the end of each PI and that serves as its capstone.

A regular time to reflect, collect data, and solve problems, the workshop is where teams and stakeholders assess the solution in process and define and take action on the improvements needed to increase the velocity, quality, and reliability of the next PI.

All program stakeholders participate, resulting in a full understanding of the current context, along with a set of improvement stories that can be added to the backlog for the upcoming PI planning. As a result, every ART improves every PI. Continuous improvement is assured with implementation of the identified backlog improvement items.

The I&A has three parts.

- *PI system demo*. This is a demo of all features completed by the ART during the previous PI.
- *Quantitative measurement*. A review of any quantitative metrics that teams have agreed to collect and discuss.
- *Problem-solving workshop*. A short retrospective for the PI, along with a structured problem-solving workshop (as shown in Figure 19-8), which focuses on identifying the root causes of the problems faced by the ART and, most importantly, pinpoints a small number of improvement items that can be added to the backlog for the upcoming PI.

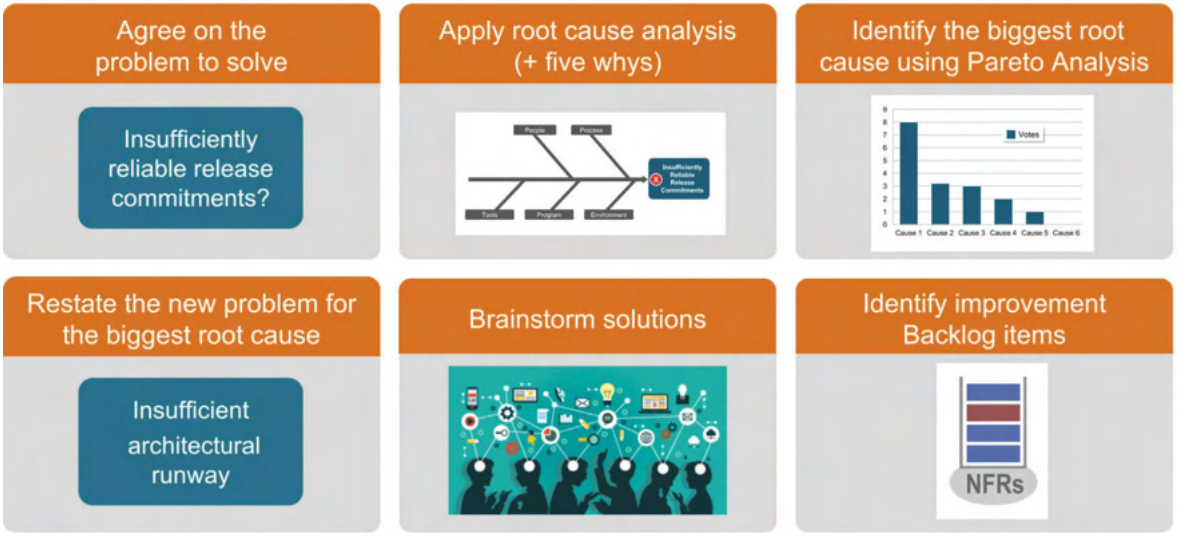


Figure 19-8. Problem-solving workshop format

What Happens without the Inspect & Adapt?

- Programs do not predictably deliver committed objectives.
- There is no way to improve systemically.
- Improvement efforts address symptoms, not root causes.
- Centralized improvement mandates don’t reflect actual development problems.

- Management stakeholders are not involved in changing the system.

IP Iteration

Every PI delivers value. During the PI, teams are busy working on the PI objectives that they committed to in PI planning. Every iteration counts, and the teams are mostly "heads down," focused on near-term delivery. There is a sense of urgency about every iteration and every PI. Given this urgency, there's a risk that if time is not put aside for innovation, improvement, and planning, the "tyranny of the urgent" will outweigh all other activities.

To address this, the IP iteration provides a regular, cadence-based opportunity for teams to work on activities that are difficult to fit into a continuous delivery model. These can include time for innovation and exploration, a dedicated time for the scheduled PI system demo, the I&A workshop, PI planning events, backlog refinement for the next PI, and even time for continuing education.

The IP iterations fulfill another critical role: They provide an estimating buffer for meeting PI objectives and enhancing release predictability. Lean teaches us that operating at near 100 percent utilization drives unpredictable results. Put simply, if everyone is planned to full capacity, there is no one available to flex when problems inevitably occur. The result is unpredictability and delays in value delivery.

To address this, the IP iteration is also treated as a "guard band" or estimating buffer. During PI planning, no features or stories are planned for development in this iteration. This buffer gives the teams extra time to respond to unforeseen events, delays in dependencies, and other issues, which increases their ability to meet PI objectives. This substantially improves the predictability of the program's outcomes, an attribute that's extremely important to Business Owners.

Routinely using that time for completing the work, however, is a failure pattern. Teams must also take care that the IP iteration does not simply become a crutch for poor planning or, worse, a time for the quality activities that must occur during the iterations themselves. It defeats the primary purpose of the IP iteration; innovation will surely suffer.

What Happens without IP Iterations?

- There is no capacity buffer, and the ARTs are not predictable.
- There is no time for innovation because of delivery urgency.
- Technical debt just grows and grows.
- People burn out.
- Cadence and schedule become a challenge as there is no time allocated for teams to plan together, demo together, and improve together.
- There is no time for continuing education.
- The real velocity slows down.

DevOps Pipeline

The goal of software and systems engineering is to deliver usable and reliable solutions to the end users. Lean and Agile both emphasize the ability to do so more frequently and reliably.

Once Agile Release Trains are launched and value streams begin to better operate, that will increase visibility into the next set of bottlenecks and impediments to improved economics through more incremental delivery. Often, "leaning out" the development cycle just moves the bottleneck further down the value stream toward deployment. The countermeasure is the implementation of the DevOps pipeline, which was covered in chapter 8, "Executing a Program Increment." Because DevOps is integral to the value stream, SAFe ARTs include deployment personnel.

What Happens without the DevOps Pipeline?

- Value delivery is greatly delayed.
- Reduced quality of deployments and a higher rate of production defects.
- More frequent releases of the solution are not possible, increasing time to market and loss of first-mover advantage.
- Large batches of code are pushed to production, resulting in production emergencies.
- There is increased friction between development and operations, limiting collaboration, learning, and cultural change.

Lean-Agile Leadership

SAFe is based on a number of modern systems and software engineering disciplines, including systems thinking, Lean product development, and Agile development. Agile provides the tools needed to empower and engage teams to achieve unprecedented levels of productivity, quality, and engagement. But a broader and deeper mindset is needed to support Lean and Agile development at scale across the entire enterprise.

For SAFe to be effective, the enterprise's leaders and managers must take responsibility for Lean-Agile adoption and success. Executives must become leaders who are trained—and become trainers—in these leaner ways of thinking and operating. Without such a change and without leadership taking responsibility for the implementation, the transformation will fail to achieve the intended benefits. That's why SAFe focuses first on a Lean-Agile leadership mindset, which includes the following:

- *Thinking Lean*. Lean thinking is represented in the SAFe "House of Lean" icon, organized around six key constructs. The "roof" represents the goal of delivering Value. The "pillars" support that goal through Respect for People and Culture, Flow, Innovation, and Relentless Improvement. Lean leadership provides the foundation on which everything else stands.
- *Embracing agility*. In addition, SAFe is built entirely on the skills, aptitude, and capabilities of Agile teams and their leaders. And while there is no one definition of what an Agile method is, the Agile Manifesto provides a unified value system that has helped inaugurate Agile methods into mainstream development.

However, thinking Lean and embracing Agile aren't enough. Leading is what's required. To this

end, Lean-Agile leaders do the actions covered in the following sections.

#1: Lead the Change

Steering an organization toward Lean and Agile behaviors, habits, and results cannot be delegated. Leaders must exhibit and communicate the urgency for change, collaboratively build a plan, understand and manage the change process, and quickly solve problems. They must have knowledge of organizational change management and take a systems view for implementing the transformation.

#2: Know the Way; Emphasize Lifelong Learning

Create an environment that promotes continuous learning and fosters formal and informal groups for learning and improvement. Strive to learn and understand new developments in Lean, Agile, and contemporary management practices.

#3: Develop People

Focus on developing people's knowledge and skills rather than on being the go-to expert or coordinator of tasks. Create a team that is jointly responsible for success.

#4: Inspire and Align with Mission

Minimize constraints. Provide an inspirational mission and vision, and eliminate demotivating rules, policies, and procedures. Build Agile teams and trains organized around value.

Understand the power of self-organizing, self-managing teams. Create a safe, failure-tolerant environment for learning, growth, and mutual influence.

#5: Decentralize Decision-Making

Establish a decision-making framework. Empower others by setting the mission, developing people, and teaching them to problem-solve.

#6: Unlock the Intrinsic Motivation of Knowledge Workers

Understand the role that compensation plays in motivating knowledge work. Change from individual to shared rewards. Create an environment of mutual influence. Eliminate any and all management processes and objectives that cause internal competition.

What Happens without Lean-Agile Leadership?

- The team has no one to learn from.
- The development processes cannot continuously improve.
- We have "Agile" development with non-Agile governance, leading back to the "iron triangle" of scope, time, and budget as well as to centralized planning and commitments.

- Lead time is long as decisions must be escalated.
- Without sufficiently powerful coalition for change, Lean-Agile transformation wanes.
- People cannot experiment, fail, and learn.
- People are over-controlled, underutilized, and demotivated.

Summary

In this chapter, we provided an overview of "Essential SAFe," which represents the indispensable elements of the Scaled Agile Framework.

The key takeaways of this chapter are as follows:

- SAFe is a framework, not a method or recipe. Therefore, some interpretation and context-specific implementation are typically required.
- Not every implementation of SAFe realizes the full business benefits that others achieve. The root causes of this problem are typically because of what happens when some essential SAFe practice is missing.
- Essential SAFe provides an excellent starting point to begin implementation of the framework. It can also serve as a diagnostic assessment for those who have already implemented SAFe.
- There are ten essential elements of SAFe:
  1. Lean-Agile principles
  2. Agile teams and ARTs
  3. Cadence and synchronization
  4. Essential team and program roles
  5. PI planning
  6. System demo
  7. I&A
  8. IP iteration
  9. DevOps pipeline
  10. Lean-Agile leadership