**SECTION: 4 :24%**

Iteration Execution

Iteration Execution is how [Agile Teams](https://www.scaledagileframework.com/agile-teams/) manage their work throughout the [Iteration](https://www.scaledagileframework.com/iterations/) timebox, resulting in a high-quality, working, tested system increment.

Developing high-quality systems is a challenge for every Agile team, [Agile Release Train (ART)](https://www.scaledagileframework.com/agile-release-train/), and [Solution Train](https://www.scaledagileframework.com/solution-train/). No matter the preparation, no matter the planning, without effective iteration execution, scaling is nearly impossible, and solution quality is compromised.

During the iteration, each team collaborates to define, build, and test the [Stories](https://www.scaledagileframework.com/story/) discussed and selected in the course of [Iteration Planning](https://www.scaledagileframework.com/iteration-planning/). They track the iteration’s progress and improve the flow of value by using story and Kanban boards and Daily Stand-up (DSU) meetings. They deliver stories throughout the iteration and avoid ‘waterfalling’ the timebox. They apply [Built-In Quality](https://www.scaledagileframework.com/built-in-quality/) practices to build the system right.

These completed stories are demoed throughout the iteration and at the [Iteration Review](https://www.scaledagileframework.com/iteration-review/). During the [Iteration Retrospective](https://www.scaledagileframework.com/iteration-retrospective/), the Agile team reflects on their practices and challenges and makes small improvements every increment. They work effectively with other teams on the train and participate in the [System Demo](https://www.scaledagileframework.com/system-demo/).

Details

Empowering Agile teams to focus on rapid value delivery fuels them with energy, motivation, and purpose. It instills a better sense of mission than traditional management and development models. The centerpiece of this approach is the developing high-quality system increments during the iteration. Teams employ a variety of practices to achieve that result, but the focus is always the same: to deliver the stories they committed to during iteration planning to meet the [Iteration Goals.](https://www.scaledagileframework.com/iteration-goals/)

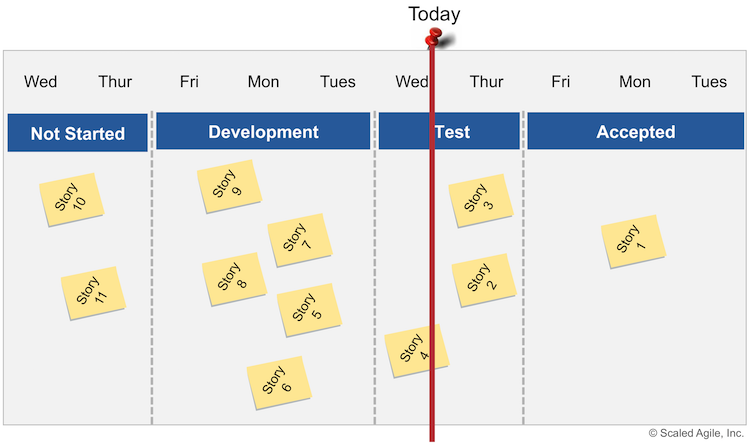
But even with good, local execution, teams are part of a larger purpose, which is optimizing program execution, one of the four [Core Values](https://www.scaledagileframework.com/safe-core-values/) of SAFe. Agile teams operate in the context of the ART, which guides teams toward the agreed-to team and Program [PI Objectives](https://www.scaledagileframework.com/pi-objectives/). All teams use the same iteration cadence and duration to synchronize their work for integration, evaluation, and demonstration during the iteration review and system demo.

The main elements of a successful iteration execution include:

* **Tracking iteration progress** – using story and Kanban boards to follow the progress of the iteration
* **Building stories serially and incrementally** – this avoids mini-waterfalls within the iteration
* **Constant communication** – continuous communication and synchronization via DSU meetings
* **Improving flow** – optimizing flow by managing Work in Process (WIP), building quality in, and continuously accepting stories throughout the iteration
* **Program execution** – working together as an ART to achieve program PI objectives

Tracking Iteration Progress

Tracking iteration progress requires visibility into the status of user stories, defects, and other team activities. For this purpose, most teams use a Big Visible Information Radiator (BVIR) on a wall in the team room. [Kanban teams](https://www.scaledagileframework.com/team-kanban/) use their Kanban board, while [ScrumXP](https://www.scaledagileframework.com/scrumxp/) teams would use a storyboard, perhaps similar to Figure 1.

Figure 1. Tracking progress with a team storyboard

With this simple storyboard, the team just moves the red ribbon to the current day, providing an easy-to-understand visual assessment of iteration progress. It’s now clear that the iteration shown in Figure 1 is at risk and the team can figure out the best way of completing the iteration. The storyboard can be shared with remote participants or stakeholders using a webcam, email, wiki, or Agile project management tooling. But this is usually in addition to the BVIR.

Constant Communication

An open work environment and collocation of team members are essential to collaboration. Otherwise, delays in value delivery will rule the day. If a team is geographically distributed, they can improve communication by leaving webcams, instant messaging, and other collaboration tools in an always-on state.

The Daily Stand-Up

Each day, the team meets at the same time and place to coordinate their work by answering the following questions:

* What did I do yesterday to advance the iteration goals?
* What will I be able to complete today to advance the iteration goals?
* What’s preventing us from completing the iteration goals?

The DSU is key to team synchronization and self-organization. It’s most useful when held in front of a BVIR that highlights the stories that are part of the team’s PI objectives. The DSU is strictly timeboxed to 15 minutes and is not a problem-solving or status meeting for management. Instead, its purpose is to coordinate the team’s activities and raise blocking issues and dependencies, many of which need to be resolved afterward. High performing teams use the DSU to find opportunities to help each other so that the entire team succeeds in delivering its committed iteration goals. The [Scrum Master](https://www.scaledagileframework.com/scrum-master/) writes down topics that need further discussion on the ‘meet after’ board. During the meet after, only the involved parties stay to talk. Ineffective DSUs are symptoms of deeper problems that require a systematic approach for resolution, which often becomes the responsibility of the Scrum Master.

Note: Although the DSU is a Scrum construct, many [Kanban Teams](https://www.scaledagileframework.com/team-kanban/) also hold a DSU in front of their Kanban board to coordinate and inspect it for bottlenecks or WIP problems.

Improving Flow

Managing WIP

WIP limits provide a strategy for preventing bottlenecks in development and helping improve flow. They also increase focus and information sharing, while fostering collective ownership. All SAFe teams should have a solid understanding of their WIP and flow.

Kanban teams explicitly apply WIP limits. ScrumXP teams may also use WIP limits. These can be explicit or implicit. For example, implicit WIP limits are when the team plans its work and takes on only the amount of stories that their capacity for the iteration predicts they can achieve. This forces the demand (negotiated iteration goals and stories) to match capacity. The iteration timebox also limits WIP by preventing uncontrolled expansion of work.

ScrumXP teams may also use explicit WIP limits on their storyboard. For example, in Figure 1 above, what would a developer do if there were no WIP limits and he or she finished story five above? The developer would probably start another story. But if a WIP limit of three is imposed on the in process and test stages, the developer would need to help test stories instead, and throughput would increase. To understand more about WIP limits, refer to [SAFe Principle #6](https://www.scaledagileframework.com/visualize-and-limit-wip-reduce-batch-sizes-and-manage-queue-lengths/).

Building Quality In

ARTs execute and deliver new functionality with the shortest sustainable lead time. But to do that, they must create high-quality systems that promote a predictable development velocity. SAFe’s [Built-In Quality](https://www.scaledagileframework.com/built-in-quality) prescribes five practices that help ensure quality at all [Solutions](https://www.scaledagileframework.com/solution/) levels:

* Achieve flow and continuous delivery by applying a test-first approach to specifying and building the system
* Apply architecture and design quality practices that create flexible systems that are easier to test and support future business needs
* Implement code quality by applying [Test-Driven-Development](https://www.scaledagileframework.com/test-driven-development/) (TDD), pair work, collective ownership, and coding standards
* Create system quality by continuously integrating the end-to-end system
* Achieve release quality through modular architectures that support smaller, more frequent component releases and an immutable infrastructure approach to changes that create more consistent, predictable releases

Ensuring that quality is built in from the beginning and at all levels makes delivering value quicker, easier, and less costly.

Continuously Accepting Stories

Accepting stories continuously improves flow. This way, problems can be addressed quickly and efficiently and avoids building new functionality on top of work that is not fit for purpose.  Further, the team avoids the context switching that otherwise occurs, when rework is required. The team reworks stories that are not accepted. Figure 1 illustrates an example of an iteration with too much WIP and insufficient flow.  After six days, the team has only moved one story to the ‘done’ state.

Test Automation

By applying [Behavior-Driven Development](https://www.scaledagileframework.com/behavior-driven-development/) (BDD), the criteria for acceptable system behavior, as specified by the [Product Owner](https://www.scaledagileframework.com/product-owner/) and the Agile team members, are converted to automated story-acceptance tests.  As the system evolves, continuously running these test helps assure that the solution previously developed and tested still performs the same way after it was changed or interfaced with other components. Automation also provides the ability to quickly regression test the system, enhancing continuous integration, refactoring, and maintenance.  Using BDD to document acceptance criteria as human readable, executable specifications encourages collaboration and helps teams keep the business goals in mind, at all times.

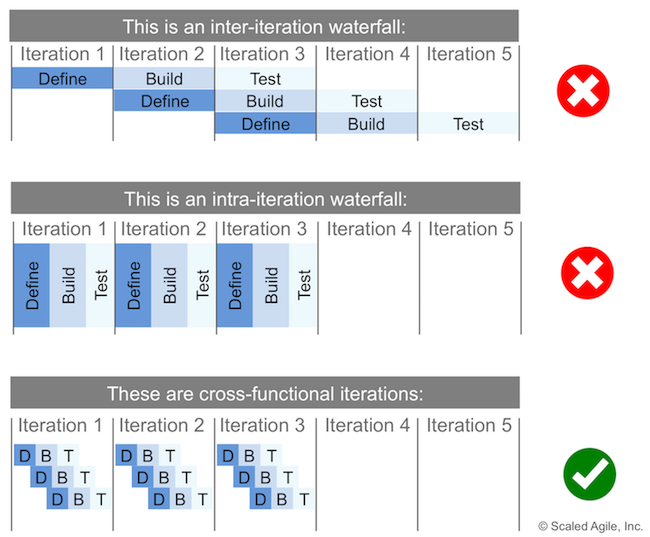
Continuous Integration and Deployment

Continuous integration at the team, system, and solution levels, as well as migrating work to a staging environment and even deploying to production allow a faster flow of value and validation of the benefit hypotheses.  The [Continuous Integration](https://www.scaledagileframework.com/continuous-integration/) and [Continuous Deployment](https://www.scaledagileframework.com/continuous-deployment/) articles further describe these practices.

Building Stories Serially and Incrementally

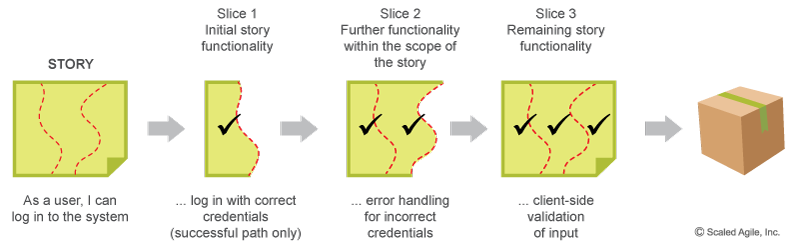
Avoiding the Intra-iteration Waterfall

Teams should avoid the tendency to waterfall the iteration and instead ensure that they are completing multiple define-build-test cycles in the course of the iteration, as Figure 2 illustrates.

Figure 2. Avoid the mini-waterfall with cross-functional Iterations

Building Stories Incrementally

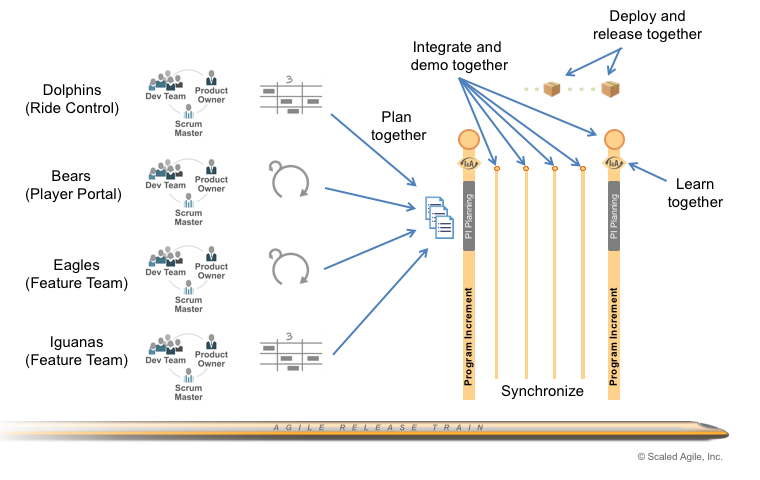
Figure 3 illustrates how implementing stories in thin, vertical slices is the foundation for incremental development, integration, and testing.

Figure 3. Implementing Stories in vertical slices is the key to incremental development

Building stories this way enables a short feedback cycle and allows the [Dev Teams](https://www.scaledagileframework.com/dev-team/) to operate with a smaller increment of the working system, allowing continuous integration and testing. It allows Dev Team members to refine their understanding of the functionality, and it facilitates pairing and more frequent integration of working systems. The dependencies within and across teams and even trains can be managed more effectively, as the dependent teams can consume the new functionality sooner. Incrementally implementing stories helps reduce uncertainty, validates architectural and design decisions, and promotes early learning and knowledge sharing.

Focusing on Program Execution

The ultimate goal of all Agile teams is the successful execution of the ART’s PI objectives. Figure 4 shows that teams: plan, integrate and demo together; deploy and release together, and learn together; to avoid focusing solely on local concerns.

Figure 4. Agile teams collaborate to achieve the program PI objectives

The [Agile Teams](https://www.scaledagileframework.com/agile-teams/) and [Dev Team](https://www.scaledagileframework.com/dev-team/) articles further describe their role in program execution.

**SECTION: 5 :20%**

Executing The Program Increment

Program Increment

A Program Increment (PI) is a timebox during which an [Agile Release Train (ART)](https://www.scaledagileframework.com/agile-release-train/) delivers incremental value in the form of working, tested software and systems. PIs are typically 8 – 12 weeks long. The most common pattern for a PI is four development Iterations, followed by one [Innovation and Planning (IP) Iteration](https://www.scaledagileframework.com/innovation-and-planning-iteration/).

A Program Increment is to an ART (or Solution Train), as an ‘[Iteration](https://www.scaledagileframework.com/iterations/) is to the [Agile Team.](https://www.scaledagileframework.com/agile-teams/)‘ It’s a fixed timebox for building and validating a full system increment, demonstrating value, and getting fast feedback. Each PI uses cadence and synchronization to:

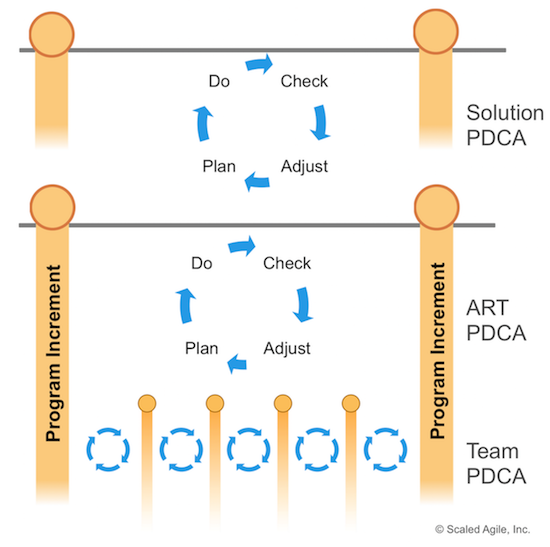
* Facilitate planning
* Limit Work in Process (WIP)
* Summarize newsworthy value for feedback
* Assure consistent, [Program Level](https://www.scaledagileframework.com/program-level/) retrospectives

Due to its scope, the PI provides several observations appropriate for [Portfolio Level](https://www.scaledagileframework.com/portfolio-level/) consideration and ‘roadmapping.’

Details

SAFe divides the development timeline into a set of [Iterations](https://www.scaledagileframework.com/iterations/) within a PI. The Big Picture illustrates how a PI is initiated by a [PI Planning](https://www.scaledagileframework.com/pi-planning/) session and is then followed by four execution iterations, concluding with one [Innovation and Planning iteration](https://www.scaledagileframework.com/innovation-and-planning-iteration/). This pattern is suggestive but arbitrary, and there is no fixed rule for how many iterations are in a PI. Experience has shown, however, that a PI duration between 8 and 12 weeks works best, with a bias toward the shortest duration.

A [Solution Train](https://www.scaledagileframework.com/solution-train/) and its associated Agile Release Trains use the same PI cadence, as shown in Figure 1.

Figure 1. The Solution Train and Agile Release Trains follow the same PI cadence

The PI represents the outer loop of the [Shewhart’s PDCA](https://en.wikipedia.org/wiki/PDCA) cycle as shown at the top of Figure 1. It combines the value developed by each [Agile Team](https://www.scaledagileframework.com/agile-teams/) into a meaningful [Milestone](https://www.scaledagileframework.com/milestones/) to objectively measure the [Solution](https://www.scaledagileframework.com/solution/) under development.

The PDCA learning cycle (shown in Figure 1) is represented by the following events in SAFe for the PI (outer loop):

* **Plan** – The [PI Planning](https://www.scaledagileframework.com/pi-planning/) event is the plan step of the cycle.
* **Do** – PI execution is the do
* **Check** – The [System Demo](https://www.scaledagileframework.com/system-demo/) is the check
* **Adjust** – The [Inspect and Adapt](https://www.scaledagileframework.com/inspect-and-adapt/) (I&A) is the adjust step

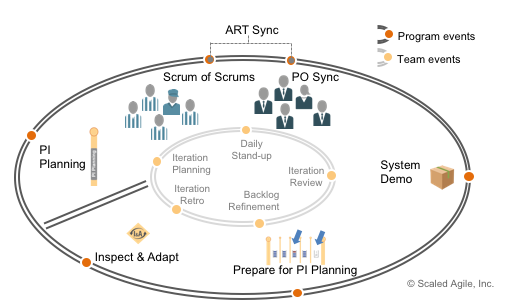
Develop on Cadence. Release on Demand

Continuous execution of PIs provides the rhythm for trains, and the assets they create grow iteratively and incrementally. Releasing solutions, however, is a separate concern, which is covered in the [Release on Demand](https://www.scaledagileframework.com/release-on-demand/) article. While trains determine the best product development rhythm, the business is enabled to deploy releases whenever it, or the market, requires.

The cadence for the PI can be different from the release cadence. However, in some situations, the PI and release cadences are the same, which can be a major convenience. Other ARTs may need to release less or more frequently than the PI cadence. Still, others will have multiple, independent release cycles for the solution’s various components.

Executing the Program Increment

When it comes to PI execution for a single ART, a sequence of program events creates a closed-loop system to keep the train on the tracks, as illustrated in Figure 2.

Figure 2. Program execution events

Each program event is described in the next sections.

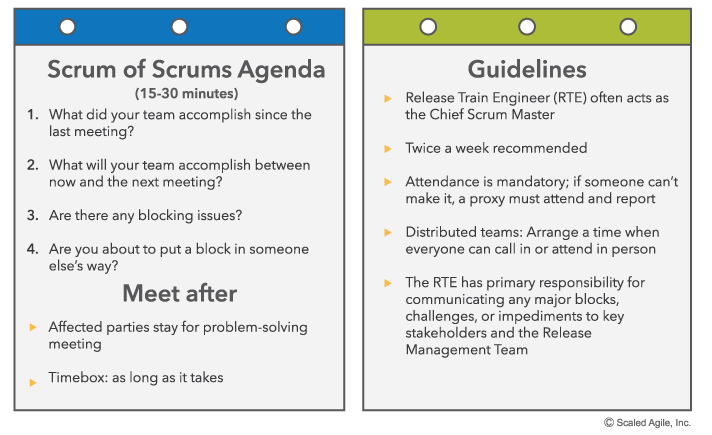
PI Planning

Each PI begins with a PI planning event. Since PI planning occurs on a fixed cadence, the entire calendar year of events can be scheduled well in advance. By scheduling PI planning events in advance, the enterprise can lower the cost of travel and logistics. It also helps people on the train, especially [Business Owners](https://www.scaledagileframework.com/business-owners/), to manage their personal calendars to assure they can be present for these critical events.

During PI planning, the teams estimate what will be delivered and highlight their dependencies with other Agile teams and trains. PI planning also creates a rhythm for the integration of work and system demos. One outcome of the PI planning is a set of program [PI Objectives](https://www.scaledagileframework.com/pi-objectives/), detailing what the ART should have ready for integration and demo at the end of the PI. Of course, Agile teams continuously integrate their work and demo it during the [Iteration Review](https://www.scaledagileframework.com/iteration-review/) and [System Demo](https://www.scaledagileframework.com/system-demo/) (or [Solution Demo](https://www.scaledagileframework.com/solution-demo/) for Solution Trains)

Scrum of Scrums

The [Release Train Engineer (RTE)](https://www.scaledagileframework.com/release-train-engineer-and-solution-train-engineer/) typically facilitates a weekly (or more frequently, as needed) Scrum of Scrums (SoS) meeting. The SoS helps coordinate the dependencies of the ARTs and provides visibility into progress and impediments. The RTE, [Scrum Masters](https://www.scaledagileframework.com/scrum-master/), and others (where appropriate) meet to review their progress toward milestones, program PI objectives, and internal dependencies among the teams. The meeting is timeboxed for less than 30 minutes and is followed by a ‘meet after’ to solve any problems. A suggested agenda for the SoS meeting is described in Figure 3.

Figure 3. Example SoS agenda

Product Owner Sync

In a manner similar to the SoS, a PO sync meeting is often held for POs and [Product Managers](https://www.scaledagileframework.com/product-and-solution-management/). This meeting typically occurs weekly, or more frequently, as needed. The PO sync is also timeboxed (30 – 60 minutes) and is followed by a meet after to solve any problems.

The PO sync may be facilitated by the RTE or Product Manager. The purpose is to get visibility into how well the ART is progressing toward meeting the program PI objectives, to discuss problems or opportunities with [Feature](https://www.scaledagileframework.com/features-and-capabilities/) development, and to assess any scope adjustments. The meeting may also be used to prepare for the next PI (see below) and may include [Program Backlog](https://www.scaledagileframework.com/program-and-solution-backlogs/) refinement and [Weighted Shortest Job First (WSJF)](https://www.scaledagileframework.com/wsjf/) prioritization before the next PI planning meeting.

Note: As illustrated in Figure 2, sometimes the SoS and PO sync are combined into one meeting, often referred to as an ART sync.

Release Management Meetings

Release management meetings provide governance for any upcoming releases, as well as communication to management. To learn more, read the [Release on Demand](https://www.scaledagileframework.com/release-on-demand/) article.

System Demo

A system demo is a biweekly event that provides feedback from the stakeholders about the effectiveness and usability of the system under development. This demo also helps ensure that integration between teams on the same ART occurs on a regular basis, no less than every iteration. And as “integration points control product development” [1], the PI is the routine point at which the meaningful, emergent behavior of the full system or solution can be evaluated.

Prepare for the Next PI Planning Event

While we note this function as an event in Figure 3, in reality, preparing for the upcoming PI is a continuous process, with three primary focus areas:

* Management alignment and organizational readiness for planning
* Backlog readiness
* The actual logistics for the event—facility readiness

Since any one of these can interfere with the potential outcome—a specific and committed PI plan—careful consideration of all three factors is necessary.

Inspect and Adapt

The PI is done when its timebox expires. Each PI is followed by a final system demo, a newsworthy event that illustrates all the [Features](https://www.scaledagileframework.com/features-and-capabilities/) that have been accomplished during the PI. This is usually done as part of the I&A workshop, which is a regular time to reflect, problem-solve, and take on improvement actions needed to increase the velocity, quality, and reliability of the next PI. The result of the workshop is a set of improvement backlog features or [Stories](https://www.scaledagileframework.com/story/) that can be added to the backlog for the upcoming PI planning. In this way, every ART improves every PI.

Solution Train PI Execution

The [Large Solution Level](https://www.scaledagileframework.com/large-solution-level/) has additional important events and activities, which bring a similar focus to the progress of the solution and are described next.

Pre- and Post-PI Planning

[Pre- and Post-PI Planning](https://www.scaledagileframework.com/pre-and-post-pi-planning/) events are used for preparation and coordination of PI planning across multiple ARTs and Suppliers in a [Solution Train](https://www.scaledagileframework.com/solution-train/). The purpose of these events is to create a common [Vision](https://www.scaledagileframework.com/vision/) and mission and a set of features that will advance the solution in alignment.

The pre-PI planning event is used to coordinate input (e.g., objectives, key milestones, business context, and [Solution Context](https://www.scaledagileframework.com/solution-context/)) for the ART planning sessions. The post-PI planning event is used to integrate the results from individual ART planning sessions into the vision and [Roadmap](https://www.scaledagileframework.com/roadmap/) for the Solution Train.

At the end of the post-PI planning meeting, there should be an agreed set of solution PI objectives to be implemented by the end of the PI and demoed at the next solution demo.

Solution Increment and Solution Demo

During the PI timebox, the ARTs build multiple increments of value, which grow into solution [Capabilities](https://www.scaledagileframework.com/features-and-capabilities/). The new capabilities must be designed, developed, tested, and validated holistically, along with the existing capabilities of the system. The solution demo is a critical aspect of the PI learning cycle. This high-profile event allows large solution stakeholders, [Customers](https://www.scaledagileframework.com/customer/) (or their internal proxies), and senior management to view the progress that the solution has made during the past PI.

At this event, the Solution Train demos its accomplishments for the entire PI. Senior managers and stakeholders review the progress in the broader solution context. It may also inform decisions about whether to pivot or persevere with capabilities, as well as changes to [Lean Budgets](https://www.scaledagileframework.com/lean-budgets/) for the various [Value Streams](https://www.scaledagileframework.com/value-streams/).

Solution Train Inspect and Adapt

At the end of the PI, an additional I&A workshop may be required for the large solution level. It follows the same format as the program level I&A. Due to the number of people involved, attendees at the Solution Train I&A cannot include all stakeholders from the ARTs, so the best-suited representatives are selected to address that context. This includes the primary stakeholders of the Solution Train, as well as representatives from the various ARTs and suppliers.

SECTION : 2

Building An Agile Team

Scrum Master

**Find a Course**:

Top of Form

  
Go

Bottom of Form

[](https://www.scaledagile.com/certification/courses/safe-advanced-scrum-master/)

Scrum Masters are servant leaders and coaches for an [Agile Team](https://www.scaledagileframework.com/agile-teams/). They help educate the team in Scrum, Extreme Programming (XP), Kanban, and SAFe, ensuring that the agreed Agile process is being followed. They also help remove impediments and foster an environment for high-performing team dynamics, continuous flow, and relentless improvement.

Although the Scrum Master role is mainly based on standard Scrum, Agile Teams—even those teams that are applying [Kanban](https://www.scaledagileframework.com/team-kanban/)—establish this position to help the team meet its goals and coordinate activities with other teams. The Scrum Master role is taken by a team member whose primary responsibility is assisting the self-organizing, self-managing team achieve its goals. Scrum Masters do this by teaching and coaching team practices, implementing and supporting SAFe principles and practices, identifying and eliminating impediments, and facilitating flow.

Details

The Scrum Master role is a unique Agile team member who spends much of her time helping other team members communicate, coordinate, and cooperate; generally, this person assists the team in meeting their delivery goals.

The Scrum Master is a servant leader who enables teams to self-organize, self-manage, and deliver via effective Lean-Agile practices. The Scrum Master supports and enforces the Scrum process and other rules that the team has agreed. The Scrum Master also helps the team coordinate with other teams on the [Agile Release Train (ART)](https://www.scaledagileframework.com/agile-release-train/) and communicates status to management as needed.

Responsibilities

An effective Scrum Master is a team-based servant leader who:

* **Exhibits Lean-Agile leadership** – Exhibits the behaviors of a [Lean-Agile Leader](https://www.scaledagileframework.com/lean-agile-leadership/) with a [Lean-Agile Mindset](https://www.scaledagileframework.com/lean-agile-mindset/). Helps the team embrace [SAFe Core Values](https://www.scaledagileframework.com/safe-core-values/), adopt and apply [SAFe Principles](https://www.scaledagileframework.com/safe-lean-agile-principles/), and implement SAFe practices.
* **Supports the team rules** – The rules of an Agile Team are lightweight, but they are rules nonetheless, and the Scrum Master is responsible for reinforcing them. These may include the rules of Scrum, [Built-In Quality](https://www.scaledagileframework.com/built-in-quality/) practices from Extreme Programming (XP), Work in Process (WIP) limits from Kanban, and any other process rules the team has agreed.
* **Facilitates the team’s progress toward team goals** – The Scrum Master is trained as a team facilitator and is continuously engaged in challenging the old norms of development to improve performance in the areas of quality, predictability, flow, and velocity. They help the team focus on daily and [Iteration Goals](https://www.scaledagileframework.com/iteration-goals/) in the context of current [Program Increment (PI) Objectives](https://www.scaledagileframework.com/pi-objectives/).
* **Leads team efforts in relentless improvement** – Helps the team improve and take responsibility for their actions; facilitates the team retrospective. Teaches problem-solving techniques and helps the team become better problem-solvers for themselves.
* **Facilitates meetings** – Facilitates all team meetings, including (where applicable) the Daily Stand-up, [Iteration Planning](https://www.scaledagileframework.com/iteration-planning/), [Iteration Review](https://www.scaledagileframework.com/iteration-review/), and [Iteration Retrospective](https://www.scaledagileframework.com/iteration-retrospective/).
* **Supports the Product Owner** – The Scrum Master helps the Product Owner in their efforts to manage the backlog and guide the team while facilitating a healthy team dynamic with respect to priorities and scope.
* **Eliminates impediments** – Many blocking issues will be beyond the team’s authority or may require support from other teams. The Scrum Master actively addresses these issues so that the team can remain focused on achieving the objectives of the [Iteration](https://www.scaledagileframework.com/iterations/).
* **Promotes SAFe quality practices** – SAFe provides guidance to assist the teams in constantly improving the quality of their deliverables and meeting the Definition of Done (DoD). The Scrum Master helps foster the culture of technical discipline and craftsmanship that is the hallmark of effective Agile teams.
* **Builds a high-performing team** – Focuses on ever-improving team dynamics and performance. Helps the team manage interpersonal conflicts, challenges, and opportunities for growth. Escalates people problems to management where necessary, but only after internal team processes have failed to resolve the issue; helps individuals and teams through personnel changes.
* **Protects and communicates** – Communicates with management and outside stakeholders; helps protect the team from uncontrolled expansion of work.
* **Responsibilities on the train** – The Scrum Master helps coordinate inter-team cooperation and helps the team operate well on the train.
* **Coordinates with other teams** – The Scrum Master is typically the representative in the Scrum of Scrums (SoS) meeting, and they pass information from that meeting back to the team (see [Program Increment](https://www.scaledagileframework.com/program-increment/) for more details). They often coordinate with the [System Team](https://www.scaledagileframework.com/system-team/), [User Experience](https://www.scaledagileframework.com/user-experience/), Architecture, and [Shared Services](https://www.scaledagileframework.com/shared-services/). It is important to note, however, that the responsibility for inter-team coordination cannot be delegated entirely to the Scrum Master; every team member shares responsibility in that regard.
* **Facilitates preparation and readiness for ART events** – Assists the team in preparation for ART activities, including [PI Planning](https://www.scaledagileframework.com/pi-planning/), [System Demos](https://www.scaledagileframework.com/system-demo/), and the [Inspect and Adapt](https://www.scaledagileframework.com/inspect-and-adapt/).
* **Supports estimating** – Guides the team in establishing normalized estimates and helps the team understand how to estimate [Features and Capabilities](https://www.scaledagileframework.com/features-and-capabilities/).

Sourcing the Role

The Scrum Master can be a part-time or full-time role, depending on the size of the team, the context, and other responsibilities. However, at [Enterprise](https://www.scaledagileframework.com/enterprise/) scale, it can be a challenge to sell the need for a full-time Scrum Master for each Agile team. After all, if the enterprise is organizing 100 new teams, it probably isn’t economically or politically practical to take 100 full-time development team members and assign them to these new duties—duties that don’t include development or testing. Nor is it economically viable to hire a full- or part-time consultant for each team to help them learn and master the new methods. That could kill the transformation before it even gets started, and before the teams have had a chance to prove the value of the role.

Therefore, SAFe takes a pragmatic approach and assumes, in general, that the Scrum Master is a part-time role. During initial SAFe adoption, however, the job may be more intensive. At this stage, the organization may find it beneficial to bring external consultants on board to coach the teams while they become experienced in Scrum and SAFe. These outside consultant Scrum Masters will often coach multiple teams in the organization.

Dev Team

**Find a Course**:

Top of Form

  
Go

Bottom of Form

[](https://www.scaledagile.com/certification/courses/safe-devops/)

The Dev Team is a subset of the [Agile Team](https://www.scaledagileframework.com/agile-teams/). It consists of dedicated professionals who can develop, test, and deploy a [Story](https://www.scaledagileframework.com/story/), [Feature](https://www.scaledagileframework.com/features-and-capabilities/), or component. The Dev Team typically includes software developers and testers, engineers, and other dedicated specialists required to complete a vertical slice of functionality.

For consistency with the Scrum definition, the Dev Team excludes the Product Owner and Scrum Master; they are part of the larger Agile Team.

Details

Developers and testers are the core of Agile development. They work in small, cross-functional teams and can quickly create working, tested code that delivers value. They build the systems we all depend on.

In traditional development, the developer and tester roles are typically differentiated, often with each reporting into a different management structure. In Agile, the two roles blend. For example, Mike Cohn doesn’t distinguish the roles and instead calls everyone “developers” [1]. Engineers that develop hardware, firmware, and other components are also considered ‘developers’ in the context of SAFe. The line between development and test is purposely blurred. Developers test and testers code.

Dev teams are empowered by the enterprise to manage and self-organize to accomplish their own work.  They include all the skills the team needs to create an increment of a working, tested solution.

Responsibilities

The Dev Team’s responsibilities include:

* Collaborating with the [Product Owner](https://www.scaledagileframework.com/product-owner/) to create and refine user stories and acceptance criteria
* Participating in [PI Planning](https://www.scaledagileframework.com/pi-planning/) and creating [Iteration](https://www.scaledagileframework.com/iterations/) plans and [Team PI Objectives](https://www.scaledagileframework.com/pi-objectives/)
* Developing and committing to [Team PI Objectives](https://www.scaledagileframework.com/pi-objectives/) and Iteration goals
* Working with the [Product Owner](https://www.scaledagileframework.com/product-owner/) to confirm that the code and acceptance tests reflect the desired functionality; writing the code
* Conducting research, design, prototyping and other exploration activities
* Creating unit tests and automated acceptance tests
* Using design and coding best practices to build high-quality components and solutions
* Checking new code into the shared source code repository
* Pairing to write code and automated acceptance test cases
* Executing acceptance tests and maintaining the test cases in a shared repository
* Continuous improvement of the team’s process

Collocation, Collaboration, and Knowledge Sharing

The collocation of Agile team members and the blurring of the traditional roles optimizes velocity and quality. It also helps create empowered Agile teams. However, this means that developers no longer operate collectively from a shared resource pool, where one could argue that it was easier to learn, share, and advance collective competencies. To address this, the Agile enterprise must consciously create a culture and environment where best practices and knowledge are shared. This includes new-found Agile skills such as story writing, [Continuous Exploration](https://www.scaledagileframework.com/continuous-exploration/), [Continuous Integration](https://www.scaledagileframework.com/continuous-integration/), [Continuous Deployment](https://www.scaledagileframework.com/continuous-deployment/), collective code ownership, and automated unit and acceptance testing, which are easily shared across teams. This is often facilitated by [Communities of Practice](https://www.scaledagileframework.com/communities-of-practice/).

Continuous Delivery

As implied above, the Dev Team is directly responsible for much of the culture and many of the practices necessary for building the [Continuous Delivery Pipeline](https://www.scaledagileframework.com/continuous-delivery-pipeline/) and implementing [DevOps](https://www.scaledagileframework.com/devops/). To this end, their ‘T-shaped’ skill training also includes developing expertise in developing and managing staging and deployment environments and mastering techniques for independently releasing elements of the larger [Solution](https://www.scaledagileframework.com/solution/). They assume additional responsibilities for following the code downstream and into production. This further blends traditional siloed responsibilities such that Agile teams, and even individual developers, can master the ability to [Release on Demand](https://www.scaledagileframework.com/release-on-demand/).

Build High-Quality Components and Solutions

Dev Teams are relentless about quality and [build quality into](https://www.scaledagileframework.com/built-in-quality) their development artifacts and practices.  They follow [test-first](https://www.scaledagileframework.com/team-and-technical-agility#testfirst) practices when writing tests for their code ([Test-Driven-Development](https://www.scaledagileframework.com/test-driven-development)) and acceptance tests for their stories ([Behavior-Driven-Development](https://www.scaledagileframework.com/behavior-driven-development)).  Agile Dev Teams apply quality design (abstraction, encapsulation, SOLID, Design Patterns) and implementation (Extreme Programming) practices so they can reliably and quickly meet new business goals. And they relentlessly [refactor](https://www.scaledagileframework.com/refactoring) their designs to ensure they can continually respond to emerging requirements.  See the Technical Agility section of SAFe’s [Team and Technical Agility](https://www.scaledagileframework.com/team-and-technical-agility) competency article for more information.

Design for Testability

Since all code is tested code, developers design and evolve the system to support testability and test automation. In practice, [designing for testability](https://www.scaledagileframework.com/design-for-testability-a-vital-aspect-of-the-system-architect-role-in-safe/) and good design are synonymous and imply modularity, low coupling, and high cohesion of layers, components, and classes. This supports the testability of any separate fragment of logic, as well as the ability to create higher, system-level, integration tests. In a similar manner, the solution should be designed to make it easy to deploy and release

SECTION 3

ITERATION PLANNING

Iteration Planning

Iteration Planning is an event where all team members determine how much of the [Team Backlog](https://www.scaledagileframework.com/team-backlog/) they can commit to delivering during an upcoming [Iteration](https://www.scaledagileframework.com/iterations/). The team summarizes the work as a set of committed [Iteration Goals](https://www.scaledagileframework.com/iteration-goals/).

Teams plan by selecting [Stories](https://www.scaledagileframework.com/story/)from the team backlog and committing to execute a set of them in the upcoming iteration. The team’s backlog has been seeded and partially planned during the [Program Increment (PI) Planning](https://www.scaledagileframework.com/pi-planning/) meeting. In addition, the teams have feedback—not only from their prior iterations but from the [System Demo](https://www.scaledagileframework.com/system-demo/) and other teams. That, and the natural course of changing fact patterns, provides the broader context for iteration planning. The output of iteration planning is:

* The iteration backlog, consisting of the stories committed to the iteration, with acceptance criteria, where appropriate
* A statement of iteration goals, typically a sentence or two stating the business objectives of the iteration
* A commitment by the team to the work needed to achieve the goals

Details

The purpose of iteration planning is to organize the work and define a realistic scope for the iteration. Each [Agile Team](https://www.scaledagileframework.com/agile-teams/) agrees on a set of [Stories](https://www.scaledagileframework.com/story/) for the upcoming iteration (the iteration backlog) and summarizes those stories into a set of iteration goals. The iteration backlog and goals are based on the team’s capacity and allow for consideration of each story’s complexity, size, and dependencies on other stories and other teams. At the end of planning, the teams commit to the goal of the iteration and adjust stories as necessary to achieve the larger purpose. In return, management does not interfere or adjust the scope of the iteration, allowing the team to stay focused on the goals.

Inputs to Iteration Planning

In SAFe, iteration planning is a refinement of the level of detail and an adjustment of the initial iteration plans created during [Agile Release Train](https://www.scaledagileframework.com/agile-release-train/) (ART) PI planning. Teams approach iteration planning with a pre-elaborated [Team Backlog](https://www.scaledagileframework.com/team-backlog/). (They have usually held a backlog refinement meeting during the previous iteration.) There are a number of inputs to the planning meeting:

* The team and [Program PI Objectives](https://www.scaledagileframework.com/pi-objectives/), created at PI planning
* The team’s PI plan backlog, which consists of stories that were identified during PI planning
* Additional stories that arise based on local context, including items such as defects, [Refactors](https://www.scaledagileframework.com/refactoring/), and new stories that have come about since the planning session
* Feedback from the prior iteration, including any stories that were not successfully completed (e.g., did not meet the ‘definition of done’) in that iteration. For more information on the definition of done, please see the “Scaled Definition of Done” paragraph in the [Built-in-Quality](https://www.scaledagileframework.com/built-in-quality/) article.
* Feedback from the system demo

Planning the Iteration

Prior to the meeting, the [Product Owner](https://www.scaledagileframework.com/product-owner/) (PO) will have prepared some preliminary iteration goals, based on the team’s progress in the [Program Increment](https://www.scaledagileframework.com/program-increment/). Typically, the Product Owner starts the meeting by reviewing the proposed iteration goals and the higher-priority stories in the team backlog. During the meeting, the Agile team discusses implementation options, technical issues, [Nonfunctional Requirements (NFRs)](https://www.scaledagileframework.com/nonfunctional-requirements/), and dependencies, then plans the iteration. The Product Owner defines the what; the team defines how and how much.

Throughout the meeting, the team elaborates the acceptance criteria and estimates the effort to complete each story. Based on their available capacity for the iteration, the team then selects the candidate stories. Some break each story down into tasks and forecast them in hours to confirm that they have the capacity and skills to complete them. Once completed, the team commits to the work and records the iteration backlog in a visible place, such as a storyboard, Kanban board or tooling. Planning is timeboxed to a maximum of four hours for a two-week iteration.

Establishing Capacity

First, the team quantifies their capacity to perform work in the upcoming iteration. Each team member determines their availability, acknowledging time off and other potential duties. This activity also takes into account other standing commitments—such as maintenance—that is distinct from new story development (see the section about capacity allocation in the [Team Backlog](https://www.scaledagileframework.com/team-backlog/) article). Using their historical velocity as a starting point, the team subtracts the unavailable time for each team member to determine the team’s actual capacity for the iteration.

Story Analysis and Estimating

Once team capacity has been established, the team backlog is reviewed. Each story is discussed, covering relative difficulty, size, complexity, technical challenges, and acceptance criteria. Teams use [Behavior-Driven Development](https://www.scaledagileframework.com/behavior-driven-development/) (BDD) to ensure a shared understanding of each story’s specific behavior. Finally, the team agrees to a size estimate for the story. There are typically other types of stories on the team backlog as well, including [Enablers](https://www.scaledagileframework.com/enablers/) that could constitute infrastructure work, refactoring, research [Spikes](https://www.scaledagileframework.com/spikes/), architectural improvements, and defects. These items are also prioritized and estimated.

Tasking Stories

Some teams break each story into tasks. As the tasks are identified, team members discuss each one: who would be the best person(s) to accomplish it, approximately how long it will take (typically in hours), and any dependencies it may have on other tasks or stories. Once all this is understood, a team member takes responsibility for a specific task or tasks. As team members commit to tasks, they reduce their individual iteration capacity until it reaches zero. Often, toward the end of the session, some team members will find themselves overcommitted, while others will have some of their capacity still available. This situation leads to a further discussion among team members to evenly distribute the work.

While breaking stories into tasks is fairly common, it is optional and not mandated in SAFe. It is mostly used by beginner teams to learn their velocity and capabilities, and then they become unnecessary, with the team planning only with stories.

Developing Iteration Goals

Once the iteration backlog is understood, the team turns their attention to synthesizing one or more iteration goals that are based on the team and program PI objectives from the PI planning session and the iteration backlog. The closer this iteration is to the PI planning session, the more likely the program objectives will remain unchanged.

Committing to Iteration Goals

When the team’s collective capacity has been reached in terms of committed stories, no more stories are pulled from the team backlog. At this point, the Product Owner and team agree on the final list of stories that will be selected, and they revisit and restate the iteration goals. The entire team then commits to the iteration goals, and the scope of the work remains fixed for the duration of the iteration.

Attendees

Attendees of the iteration planning meeting include:

* The Product Owner
* The [Scrum Master](https://www.scaledagileframework.com/scrum-master/), who acts as the facilitator for this meeting
* The Development Team
* Any other stakeholders, including representatives from different Agile teams or the ART, and subject matter experts

Agenda

An example agenda for iteration planning follows:

1. Calculate the available team capacity for the iteration.
2. Discuss each story, elaborate acceptance criteria, and provide estimates using story points.
3. Planning stops once the team runs out of capacity.
4. Determine and agree on the iteration goals.
5. Everyone commits to the goals.

Acceptance criteria are developed through conversation and collaboration with the Product Owner and other stakeholders. Based on the story estimates, the Product Owner may change the ranking of the stories. Optionally, the team may break stories into tasks, estimated in hours, and takes shared responsibility for execution.

Guidelines

Below are some tips for holding an iteration planning meeting:

* Timebox the meeting to 4 hours or less
* This planning session is organized by the team and is for the team
* A team should avoid committing to work that exceeds its historical velocity

Relative Estimating, Velocity, Capacity, and Normalizing Story Point Estimating

Agile teams use story points to relatively estimate user stories in story points [2, 3]. With relative estimating, the size (effort) for each backlog item is compared to other stories. For example, an eight-point story is four times the effort as a two-point story. The team’s velocity is equal to the historical average of all the stories completed per iteration. Velocity is the starting point for calculating a team’s capacity for a future iteration. Knowing a team’s capacity assists with planning and helps limit Work in Process (WIP)—teams don’t take on more stories than their prior velocity would allow. Velocity is also used to estimate how long it takes to deliver [Features](https://www.scaledagileframework.com/features-and-capabilities/) or [Epics](https://www.scaledagileframework.com/epic/), which are also forecasted in story points.

Normalizing Story Point Estimating

In Scrum, each team’s story point estimation—and the resultant velocity—is typically a local and independent matter. The fact that a small team might estimate in such a way that their velocity is 50, while another larger team estimates that their velocity is 13, usually does not pose a concern.

In SAFe, however, story point estimation must be normalized, so that estimates for [Features](https://www.scaledagileframework.com/features-and-capabilities/) or [Epics](https://www.scaledagileframework.com/epic/) that require the support of multiple teams are based on the same story point definition, allowing a shared basis for economic decision-making. One starting algorithm is as follows:

**1. Normalize story point estimation:**

* Find a small story that would take about a half-day to develop and a half-day to test and validate, and call it a “one”
* Estimate every other story relative to that “one”

**2. Establish velocity before historical data exists:**

* For every full-time developer and tester on the team, give the team 8 points (adjust for part-timers)
* Subtract one point for every team member vacation day and holiday in the iteration

**Example**: Assuming a six-person team composed of three developers, two testers, and one PO, with no vacations, then the estimated initial velocity equals 5 × 8 points or 40 points per iteration. (Note: The team may need to adjust a bit lower if one of the developers and testers is also the Scrum Master.)

Normalized estimation is particularly helpful in initial PI planning, as many teams will be new to Agile and will need a way to estimate the scope of work in their first PI.  Note: There is no need to recalibrate team estimating or velocities after that point. It is just a common starting point. In this way, all teams estimate the size of work in a common fashion, so management can quickly estimate the cost for a story point for teams in a specific region. They then have a meaningful way to establish the aggregate cost estimate for an upcoming feature or epic.

While teams will tend to increase their velocity over time—and that is a good thing—in fact, the number tends to remain stable, and a team’s velocity is far more affected by changing team size, makeup, and technical context than by productivity changes. And, if necessary, financial planners can adjust the cost per story point a bit. This is a minor concern compared to the wildly differing velocities that teams of comparable size may have in the non-normalized case.