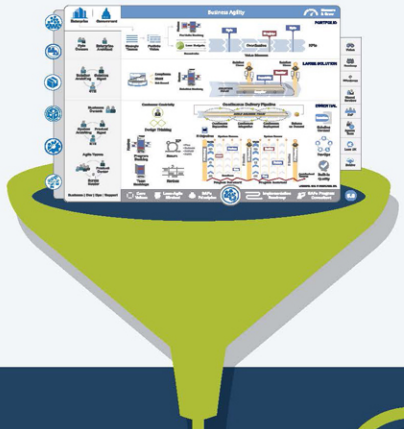
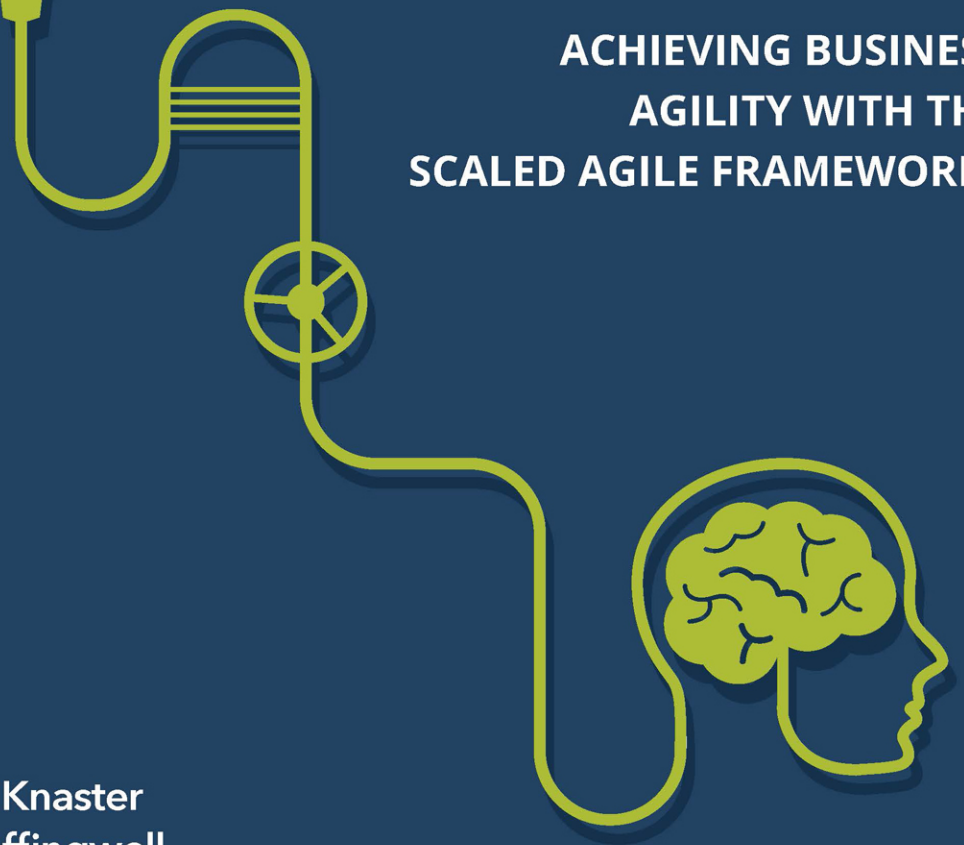
SAFe® 5.0



SAFe® DISTILLED

ACHIEVING BUSINESS AGILITY WITH THE SCALED AGILE FRAMEWORK®



Richard Knaster
Dean Leffingwell

SAFe® 5.0 Distilled

Team Backlog

The *team backlog* contains user and enabler *stories*. It may include other work items as well, representing all the things a team needs to do to advance their portion of the system.

User stories are the primary means of expressing needed functionality. Because they focus on the user as the subject of interest, and not the system, user stories are *value-centric*. To support this, the recommended form of expression is the user-voice format, shown here:

> As a (user role),
>
> I want (activity) to,
>
> so that (business value)

This user-voice format guides teams to understand who is using the system, what they are doing with it, and why they are doing it. Applying this format tends to increase the team's domain competence; they begin to better grasp the real business needs of their user. Stories may come from the team's local context; however, they also typically result from splitting business and enabler features (Figure 6-3). Features are further described in Chapter 7, Agile Product Delivery.

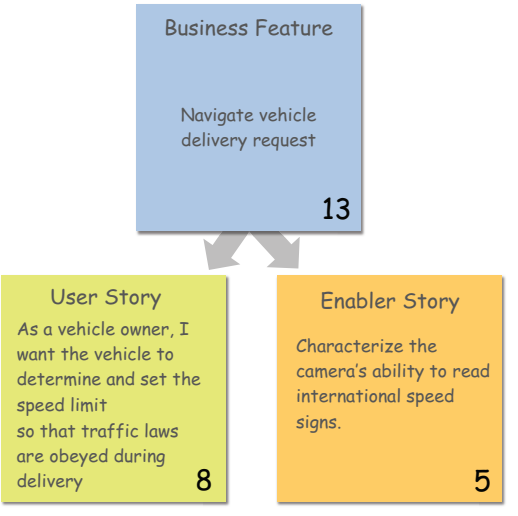


Figure 6-3. Stories typically originate from splitting business and enabler features

In addition, enabler stories describe the work needed to build the architectural runway, which supports the efficient development and delivery of future business features.

These may include items such as exploration, architecture, refactoring, infrastructure, and compliance concerns.

To achieve flow, the team backlog must always contain some stories that are ready to be implemented without significant risk or delay. This is accomplished by frequently refining the backlog. Backlog refinement looks at upcoming stories (and features, as appropriate) to discuss, estimate, and establish an initial understanding of acceptance criteria.

Also, as multiple teams refine their respective backlogs, new issues, dependencies, and stories are likely to emerge. In this way, backlog refinement helps surface problems of understanding or challenges with the current plan.

SAFe Teams Typically Blend Agile Methods

SAFe teams use Agile practices of choice based primarily on Scrum, Kanban, and quality practices derived, in part, from Extreme Programming (XP) (Figure 6-4).

Scrum is a lightweight, team-based process that fosters fast feedback and quick, iterative development of the solution. In Scrum, teams define, build, test (and where applicable, deploy) functionality in short sprints (iterations).

To assure throughput and continuous flow, most teams integrate the best practices of Kanban with Scrum to visualize their work, establish Work In Process (WIP) limits, and illustrate bottlenecks and opportunities for improving throughput. Teams whose work is more active and demand-based often choose Kanban as their primary practice. However, they still plan in cadence with other teams and typically apply the Scrum Master and Product Owner roles (or equivalents) for consistent operation within the ART.

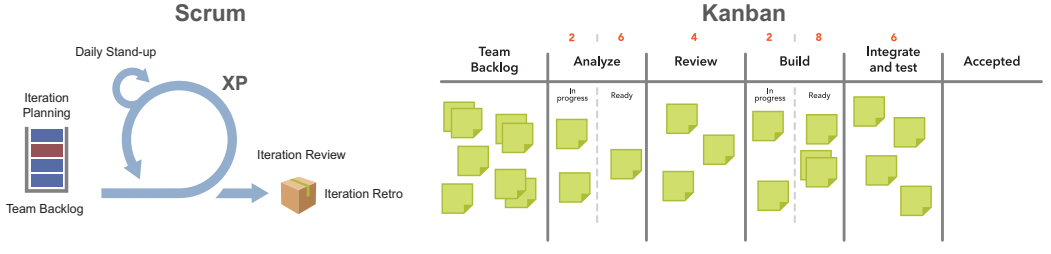


Figure 6-4. SAFe teams typically blend Agile methods

Estimating Work

Agile teams use story points to estimate their work. A story point is a single number that represents a combination of qualities.

- ***Volume***. How much is there?
- ***Complexity***. How hard is it?
- ***Knowledge***. What's known?
- ***Uncertainty***. What's unknown?

Story points are relative, without a connection to any specific unit of measure. The size (effort) of each story is estimated relative to other stories, with the smallest story assigned a size of one. A modified Fibonacci sequence (1, 2, 3, 5, 8, 13, 20, 40, 100) is applied to reflect the inherent uncertainty in estimating, as the size get larger. (e.g., 20, 40, 100).[1]

Iterating

Agile teams work in iterations, which provide a regular, predictable planning, development, and review cadence. This ensures that a full Plan–Do–Check–Adjust (PDCA) cycle is executed as quickly as possible. Each iteration is a standard, fixed timebox, which is typically one to two weeks.

These short time periods help the team and other stakeholders regularly test and evaluate the technical and business hypotheses in a working system. Teams integrate their code frequently throughout the iteration. Each iteration also anchors at least one significant, system-level integration point. This event, known as the system demo, assembles various system aspects—functionality, quality, alignment, and fitness for use—across all the teams' contributions.

Planning the Iteration

The iteration starts with iteration planning, a timeboxed event of four hours or less (for a two-week iteration). During planning, the team does the following:

- Reviews, refines, and estimates stories, which are typically presented by the PO

1. Mike Cohn, *User Stories Applied: For Agile Software Development* (Addison-Wesley, 2004).

- Defines the acceptance criteria
- Splits larger stories into smaller ones where necessary
- Determines what they can deliver in the upcoming iteration, based on their known velocity (story points per iteration), into iteration goals
- Commits to a short set of iteration goals

Some teams further divide stories into tasks, estimating them in hours to better refine their understanding of the work ahead.

Even before iteration planning starts, Agile teams prepare content by refining the team backlog. Their objective is to better understand the work to be delivered in the upcoming iteration.

Coordinating with Daily Stand-Up Events

Each day, the team has a Daily Stand-Up (DSU) to understand where they are, escalate problems, and get help from other team members. During this event, each team member describes what they did yesterday to advance the iteration goals, what they are going to work on today, and any blocks they are encountering. The DSU should take no more than 15 minutes and typically occurs standing before the team's Kanban board (or electronic equivalent for distributed teams).

But team communication does not end there, as team members interact continuously throughout the iteration. Facilitating such communication is the main reason why teams should be collocated whenever possible.

Delivering Value

During the iteration, each team collaborates to define, build, and test the stories they committed to during iteration planning, resulting in a high-quality, working, tested system increment. They deliver stories throughout the iteration and avoid 'waterfalling' the timebox. These completed stories are demoed throughout the iteration. Teams track the iteration's progress and improve the flow of value by using Kanban boards and the DSU.

To ensure they are solving the right problem, teams apply design thinking and customer centricity (see Chapter 7). To build the system right, teams also apply built-in quality practices, which are described later in this chapter.

Improving the Process

At the end of each iteration in Scrum, the team conducts an iteration review and an iteration retrospective. During the iteration review, the team's increment of stories is demoed for that iteration. This is not a formal status report; rather, it's a review of the tangible outcomes of the iteration. Teams also conduct brief retrospectives—a time to reflect on the iteration, the process, things that are working well, and current obstacles. Then the team comes up with improvement stories for the next iteration.

Teams of Agile Teams

The second dimension of the team and technical agility competency is *teams of Agile teams*. Even with good, local execution, building enterprise-class solutions typically requires more scope and breadth of skills than a single Agile team can provide. Therefore, Agile teams operate in the context of an ART, which is a long-lived team of Agile teams. The ART incrementally develops, delivers, and (where applicable) operates one or more solutions (Figure 6-5).

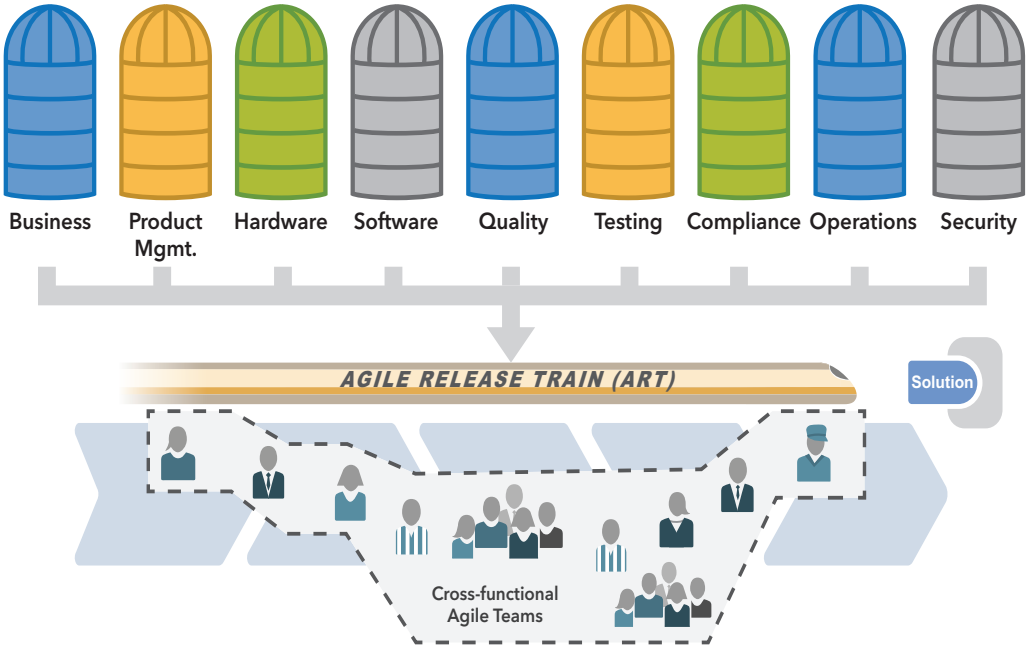


Figure 6-5. Agile Release Trains develop, deliver, and support one or more solutions

ARTs align teams to a common business and technology mission. Each is a virtual organization (typically 50 to 125 people) organized around the enterprise's significant value streams and existing solely to realize the promise of that value by building solutions that deliver benefit to the end user.

The ART applies systems thinking (SAFe Principle #2) and organizes around value (SAFe Principle #10) to build a cross-functional organization optimized to facilitate value flow from ideation through deployment and release and into operations. This creates a far leaner organization, one where traditional daily task and project management is no longer required. Value flows more quickly, with a minimum of overhead.

In addition to the Agile teams, the following roles help ensure successful ART execution:

- ***Release Train Engineer (RTE)*** is a servant leader who facilitates program execution, impediment removal, risk and dependency management, and continuous improvement.
- ***Product Management*** is responsible for 'what gets built,' as defined by the vision, roadmap, and new features in the program backlog. They are responsible for defining and supporting the building of desirable, feasible, viable, and sustainable products that meet customer needs over the product-market lifecycle.
- ***System Architect/Engineering*** is an individual or team that defines the overall architecture of the system. They work at a level of abstraction above the teams and components and define Non-Functional Requirements (NFRs), major system elements, subsystems, and interfaces.
- ***Business Owners*** are key stakeholders of the ART and have ultimate responsibility for the business outcomes of the train.
- ***Customers*** are the ultimate recipients of the solution's value.

In addition to these critical ART roles, the following functions can often play an essential part in ART success:

- ***System Teams*** typically assist in building and supporting DevOps infrastructure for development, continuous integration, automated testing, and deployment into the staging environment. In larger systems they may do end-to-end testing, which cannot be readily accomplished by individual Agile teams.