(/)

> *Specifically, you can take the time to develop and bring to the table an outside-in, market-centric perspective that is so compelling and so well informed that it can counterbalance the inside-out company-centric orientation of last year's operating plan.*
>
> —Geoffrey Moore, Escape Velocity

Continuous Exploration

Continuous Exploration (CE) is the process of constantly exploring market and user needs, and defining a Vision (/vision/), Roadmap (/roadmap/), and set of Features (/features-and-capabilities/) that address those needs. It is the first element in the four-part Continuous Delivery Pipeline (/continuous-delivery-pipeline/), preceding Continuous Integration (/continuous-integration/) (CI) Continuous Deployment (/continuous-deployment/) (CD), and Release on Demand (/release-on-demand/).

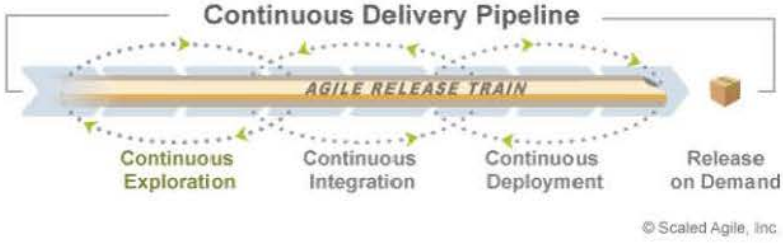


Figure 1. Continuous Exploration is the first element of the Continuous Delivery Pipeline

New features are initially captured and defined during the CE process. When each feature is ready, it is entered into the Program Backlog (/program-and-solution-backlogs/), and the continuous integration process pulls the highest priority features into implementation. Thereafter, the continuous deployment cycle pulls the features into the staging or deployment environment, where they are validated and made ready for release.

Inputs to CE come from Customers (/customer/), Agile Teams (/agile-teams/), Product Owners (/product-owner/), Business Owners (/business-owners/) as well as stakeholders, and portfolio concerns. Under the direction of Product Management (/product-and-solution-management/), various research and analysis activities are used to further define and evaluate the feature. The result of this process is a set of outputs, including the vision (/vision/), a set of features in the backlog (/program-and-solution-backlogs/) sufficiently defined for implementation, and a preliminary roadmap (/roadmap/) forecast of how those features might be delivered over time.

Details

SAFe is optimized to deliver a continuous flow of value to the customer. As a result, it avoids the traditional waterfall approach, eliminating extensive up-front definition of the work to be done. Instead, it applies a continuous exploration process, providing a consistent flow of new work that is sufficiently ready for the teams to implement. This way, new functionality is available in small batches that can travel easily through continuous integration, continuous deployment, and

on to release.

Continuous Exploration Process

In the large, the CE process may be considered to consist of three separate activities—collaboration, research, and synthesis—as can be seen in Figure 2.

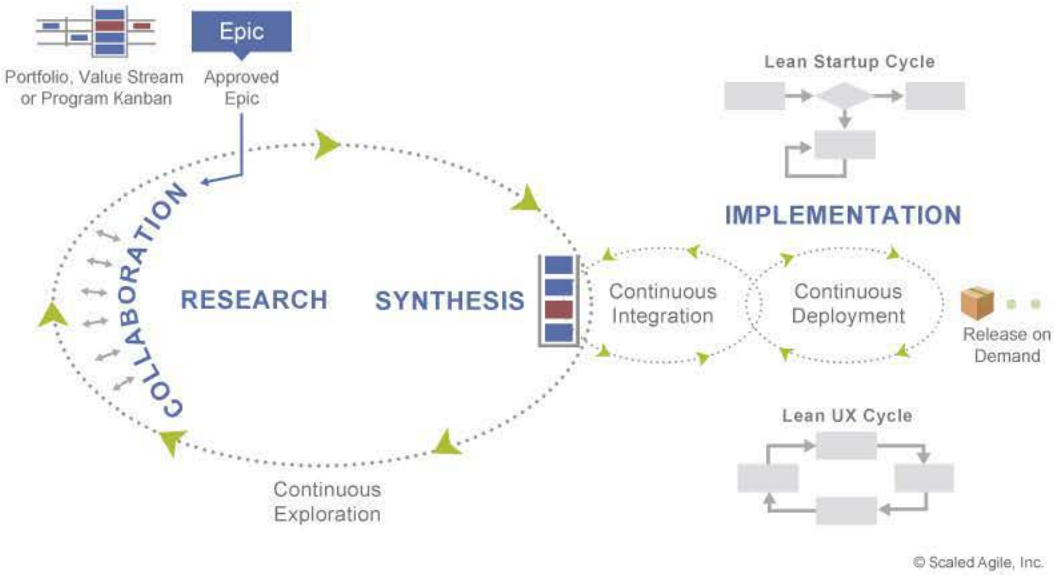


Figure 2. Continuous Exploration: collaboration, research, synthesis

Collaboration

To create a compelling and differentiated vision, Product Management enables and facilitates a continuous and collaborative process that solicits input from a diverse group of stakeholders. Primary sources include:

- **Customers** – By voting with their wallets or their feet, customers (/customer/) are the ultimate judges of value. They're the most obvious and primary source of input. But a note of caution: Customers are heavily bound to their current Solution Context (/solution-context/), so they are often motivated only to improve things incrementally. In other words: The sum total of customer input does not a strategy make. But failing to meet real and evolving customer needs is an alternate path to extinction. A sense of balance is required. As the SAFe Lean-Agile Mindset (/lean-agile-mindset/) says, "Producers innovate. Customers validate."
- **Agile teams** – The Agile teams (/agile-teams/) and product owners (/product-owner/) have some of the foremost expertise in the domain. In most cases, they developed the existing solution and are closest to both technical and user concerns. Their input is integral and invaluable.
- **Business owners** – Business owners (/business-owners/) have the business and market knowledge needed to set the mission and vision. They also have specific SAFe responsibilities throughout the development process. A solution that doesn't meet their expectations is probably no solution at all.
- **Portfolio concerns** – As we described above, SAFe Value Streams and Agile Release Trains (ARTs) operate in a larger portfolio context. Strategic Themes (/strategic-themes/) drive new Epics to improve portfolio differentiation.
- **System Architects/Engineers** – System and Solution Architects/Engineers (/system-and-solution-architect-engineering/) have in-depth technical knowledge of the solution and are responsible for understanding it at the system level, as well as its use cases and Nonfunctional Requirements (NFRs). So, although it's natural to view these roles as technically and internally inclined, the most capable—in our experience—also have significant and ongoing customer engagement.

Research

In addition to this direct input, Product Management uses a variety of research activities and techniques to help establish the vision. These may include:

- **Customer visits** – There's no substitute for first-person observation of the daily activities of the people doing the work. Whether structured or informal, Product Managers and Product Owners are responsible for understanding how people actually use systems in their actual work environments. They can't do that at their desk, so there is no substitute for observing users in their specific solution context (/solution-context/).
- **Gemba walks** – Many times, customers are the internal people who implement the operational values streams that our development systems support. In this case, the Gemba walk ('Gemba' is the place where the work is performed [4]) can be

used by developers to observe how these stakeholders execute the steps and specific activities in their operational value streams.

- **Elicitation** – There are a variety of structured elicitation techniques that Product Management and Product Owner professionals use to generate input and prioritize user needs. These include research methods such as interviews and surveys, brainstorming and idea reduction, questionnaires, and competitive analysis. Other techniques include requirements workshops, user experience mock-ups, user personas, customer change request systems, and use-case modeling. [2, 3]
- **Trade studies** – Teams engage in trade studies to determine the most practical characteristics of a solution. They review numerous solutions to a technical problem, as well as vendor-provided products and services that address the subject area or an adjacent need. Solutions are evaluated against desirable outcomes, agreeing on a hypothesis for the effective solution for a particular context.
- **Market research** – To expand their thinking, teams also conduct original market research, analyze secondary research and market/industry trends, identify emerging customer segments, interview industry analysts, and review competitive solutions.

Synthesis—Building the Vision, Roadmap, and Program Backlog

Based on this collaboration and research, Product Management professionals synthesize their findings into the key SAFe artifacts of vision, roadmap, and program backlog (/program-and-solution-backlogs/). The Program Kanban (/program-and-solution-kanbans/) helps manage this work. The default states of funnel, analysis, and backlog are good starting points to establish that workflow. The net result is a set of features that are in the backlog and are ready for implementation. These are stored in the backlog until they're ready to appear as a Weighted Shortest Job First (WSJF)– (/wsjf/)prioritized feature at Program Increment (PI) Planning (/pi-planning/).

Define Features with Lean UX Thinking

In the Lean User Experience (UX) (/lean-ux/) article, we described an approach to implementing features that followed a simple, four-step lean process model: 1. Define an outcome hypothesis; 2. Design collaboratively; 3. Implement a Minimum Marketable Feature (MMF); and 4. Evaluate the MMF against the hypothesis.

And while this model was developed and described in the context of user-facing functionality, it isn't reserved exclusively for that; all features benefit from this approach. For example, a feature such as "In service software update," might not be user facing at all, but when ready in the program backlog (/program-and-solution-backlogs/), it could appear as in Figure 3.

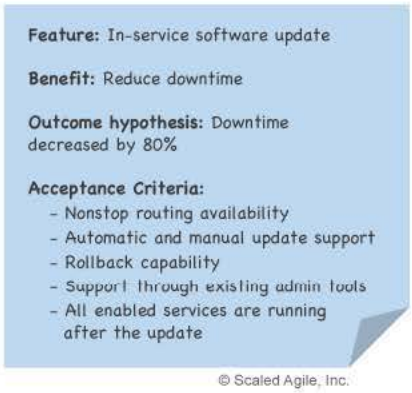


Figure 3. A ready for implementation feature statement

Implementing

That's the transition point. With a solid feature definition in hand, the next process, continuous delivery, pulls features from the backlog and implements them. In accordance with Lean UX (/lean-ux/) process [5], each MMF is collaboratively designed and developed, delivered incrementally, and objectively measured, with each feature investment tuned to optimum outcome.

Learn More

[1] Ries, Eric. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses.* Random House, Inc. Kindle Edition.

[2] Leffingwell, Dean. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series).* Pearson Education. Kindle Edition.

[3] http://www.innovationgames.com (http://www.innovationgames.com)

[4] Womack, Jim. *Gemba Walks Expanded 2nd Edition*. Lean Enterprise Institute, Inc. Kindle Edition.

[5] Gothelf, Jeff and Josh Selden. *Lean UX: Designing Great Products with Agile Teams*. O'Reilly Media. Kindle Edition.

Last update: 13 September, 2017

FRAMEWORK

Downloads (/posters/)

Latest Updates (/blog/)

TRAINING

Course Calendar (http://www.scaledagileacademy.com/events/event_list.asp)

About Certification (http://www.scaledagileacademy.com/?page=WhichCertification)

Become a Trainer (http://www.scaledagileacademy.com/?page=becomeatrainer)

PERMISSIONS

Permission FAQ (http://www.scaledagile.com/permission-faq/)

Permissions Form (http://www.scaledagile.com/permissions-form/)

Usage and Permissions (/usage-and-permissions/)

PARTNER

Becoming a Partner (http://www.scaledagile.com/become-a-partner/)

Partner Directory (http://www.scaledagile.com/listingcategory/directory/)

Partner Event Calendar (http://www.scaledagile.com/event-list/)

GET SOCIAL

Twitter (https://twitter.com/ScaledAgile)

Linkedin (https://www.linkedin.com/grps/Scaled-Agile-Framework-4189072?)

YouTube (https://www.youtube.com/user/scaledagile)

SlideShare (http://www.slideshare.net/ScaledAgile)

RECENT POSTS

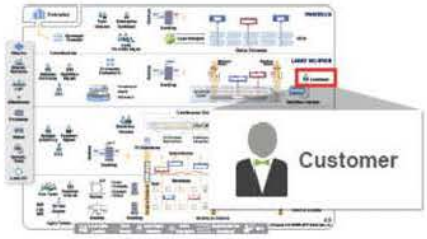
Updated SAFe Scrum Master and SAFe Advanced Scrum Master courses now available! (/blog/updated-safe-scrum-master-and-safe-advanced-scrum-master-courses-now-available/)

Sep, 19th 2017

SAFe from the Trenches at Agile Australia (/blog/safe-from-the-trenches-at-agile-australia/)

Sep, 19th 2017

Updated SAFe Release Train Engineer and SAFe Product Owner/Product Manager courses now available! (/blog/updated-safe-release-train-engineer-and-safe-product-ownerproduct-manager-courses-now-available/)



(/)

> *There is only one boss. The customer. And he can fire everybody in the company from the chairman on down, simply by spending his money somewhere else.*
> *—Sam Walton, Walmart founder*
>
> "Even when they don't yet know it, customers want something better, and your desire to delight customers will drive you to invent on their behalf. No customer ever asked Amazon to create the Prime membership program, but it sure turns out they wanted it, and I could give you many such examples.
> —Jeff Bezos
>
> Get closer than ever to your customer.
> —Steve Jobs

Customer

Customers are the ultimate economic buyer of every Solution (/solution/). They are an integral part of the Lean-Agile development process and Value Stream, and have specific responsibilities in SAFe.

Customers—whether internal or external—are increasingly demanding. They have choices. They expect solutions to work well and to solve their current needs. They also expect their solution providers to continuously improve the quality of their products and services.

Moreover, *engaged* Customers are integral to Lean-Agile solution development. They are integral to the Solution, and inseparable from the development process. They have specific responsibilities and work frequently and closely with Solution and Product Management (/product-and-solution-management/), and other key stakeholders to shape the Solution Intent (/solution-intent/), Vision (/vision/), and the Economic Framework (/economic-framework/) in which development occurs. They exert a strong influence on defining and prioritizing the solution's development, and are active participants in PI Planning (/pi-planning/), System (/system-demo/) and Solution Demos (/solution-demo/), and Inspect and Adapt (/inspect-and-adapt/) (I&A).

Details

Customers are an integral part of Lean-Agile development and play a critical role in SAFe. They are part of the Value Stream (/value-streams/). Their support for Lean and Agile Principles (/safe-lean-agile-principles/) and their active and continuous participation in the Solution (/solution/) definition, planning, demonstrations, and evolution are essential to successful execution.

In some cases, the Customer is internal (example: an IT shop delivering a supply chain application to the business). In others, the Customer is external and is the buyer of a custom-built offering by the system builder (example: government purchasing a commercial or defense system). In still others, the Customer is a more remote third party, one of a larger class of economic buyers. There, the system builder must understand the aggregate and synthesize requirements of the general case, craft solutions that fill the broader market needs, and provide an adequate internal proxy for much of development (example: an independent software vendor selling a suite of products).

Summary of Responsibilities

No matter the type, Customers must be engaged continuously throughout Agile solution development. They participate either in person or by proxy to fulfill the following general responsibilities:

- Participate as Business Owner (/business-owners/) in PI Planning (/pi-planning/)
- Attend Solution (/solution-demo/) and possibly System Demo (/system-demo/); help evaluate the solution increment
- Participate in Inspect and Adapt (/inspect-and-adapt/) workshops; assist in removing some systemic impediments
- Interact with analysts and subject matter experts during specification workshops
- Collaboratively manage scope, time, and other constraints with Product and Solution Management (/product-and-solution-management/)
- Help define the Roadmap (/roadmap/), Milestones (/milestones/), and Releases (/release-on-demand/)
- Communicate the economic logic behind the solution and help validate assumptions in the Economic Framework (/economic-framework/)
- Review technical and financial status of the solution
- Participate in beta testing, UAT, other forms of solution validation

The Customer Is Part of the Value Stream

The Lean-Agile Mindset (/lean-agile-mindset/) spans beyond the development organization to encompass the entire value stream, which includes the Customer. The type of the value stream determines the context for interaction:

1) In the case of *Internal IT*, the Internal Customer is part of the *operational value stream*, as Figure 1 illustrates.

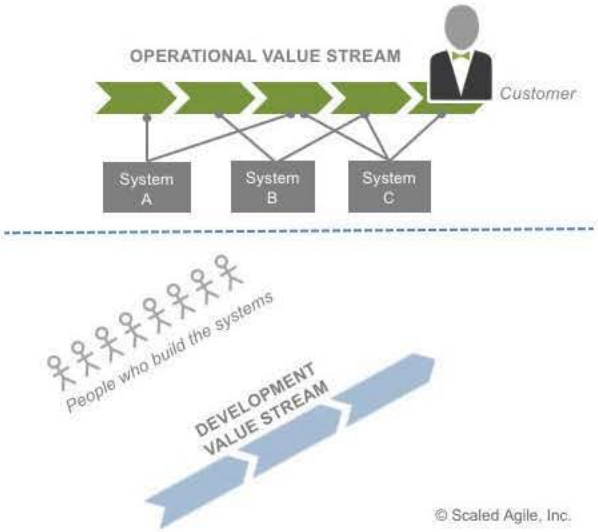


Figure 1. The Internal Customer

An example would be a marketing director who has responsibility for a partner enrollment work flow (the operational value stream). The partner is the ultimate end user of the work flow and is the Customer, but to the development team, the marketing director and those who operate the value stream are the Customer.

2) In the case of those who build solutions for an external end user, the Customer is the direct economic buyer of the solution, as Figure 2 illustrates.

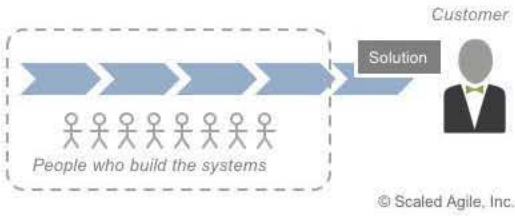


Figure 2. External Customers are direct economic buyers

In this case, the development value stream and the operational value stream are the same. The solution can be a final product that is sold or deployed directly, or it may need to be embedded into a broader Solution Context (/solution-context/), such as a system of systems, to make it operational.

Customer Engagement Drives Agile Success

Lean-Agile development is dependent on a high degree of Customer engagement, much higher than our former stage-gated models assumed. However, the means of engagement are different, based on whether the solution builder is building a *general solution*—one that can be used by or sold to a significant number of Customers—or whether the solution is a *custom built solution*—one that is built specifically for an individual Customer to their specifications. Figure 3 illustrates the relative level of indirect or direct Customer engagement in each case.

Figure 3. Customer engagement models in general and custom built solutions

General Solutions

On the left side, solution builders build systems that must address the needs of a larger audience. No one customer can be assumed to be an adequate proxy for the market as a whole. In this case, Product and Solution Management serves as the indirect Customer proxy, and they have the authority over solution content. It is their responsibility to facilitate external interaction and make sure that the voice of the Customer will be heard, and that the organization continuously validates new ideas. Scope, schedule, and budget for development are generally at the discretion of the solution builder.

Since it is unlikely that any particular Customer will be participating in regular planning and demo sessions, interaction is typically based on requirements workshops, focus groups, usability testing, innovation accounting, limited beta releases, etc. By applying user behavior analysis, measures, and business intelligence to validate the various hypotheses, the solution evolves based on this feedback. During PI planning, a group of internal and external stakeholders acts as the Business Owners, the ultimate internal Customer proxy within a specific value stream.

Custom Built Solutions

On the right side of Figure 3, the Customer is typically "in charge." Such Customers define the solution and represent themselves. Product and Solution Management interact with the Customer and provide daily development support. However, even though the Customer is in charge, it is critical to establish a collaborative approach to scope and prioritization, both to foster incremental learning and to exhibit a willingness to adjust the course of action as facts dictate.

Active participation in PI planning, the Solution Demo (/solution-demo/), and selected specification workshops is required. This will often reveal inconsistencies in requirements and design assumptions, with potential contractual ramifications. This process should drive the Customer and solution builder toward a more collaborative and incremental approach.

Demonstrating results of the Program Increment (/program-increment/) to the Customer—in the form of a fully integrated solution increment—establishes a high degree of trust ("these teams can really deliver") and also provides Customers with the opportunity to empirically validate the current course of action. Forecasting ability, based on the measured predictability and velocity of the Agile Release Trains (/agile-release-train/), is significantly improved.

Transition toward an Agile contract model will also help reduce the win-lose aspects of traditional relationships between systems builder and Customer. One such model is the SAFe managed investment model, whereby the Customer commits the funding for a PI or two, then adjusts funding based on objective evidence and incremental deliveries. This requires a fair bit of trust going in, but thereafter trust is built incrementally, based on a continuous flow of value received.

Learn More

[1] Ward, Allen and Durward Sobek. *Lean Product and Process Development.* Lean Enterprise Institute, 2014.

Last update: 9 June, 2017

FRAMEWORK

Downloads (/posters/)

Latest Updates (/blog/)

TRAINING

Course Calendar (http://www.scaledagileacademy.com/events/event_list.asp)

About Certification (http://www.scaledagileacademy.com/?page=WhichCertification)

Become a Trainer (http://www.scaledagileacademy.com/?page=becomeatrainer)

PERMISSIONS

Permission FAQ (http://www.scaledagile.com/permission-faq/)

Permissions Form (http://www.scaledagile.com/permissions-form/)

Usage and Permissions (/usage-and-permissions/)

PARTNER

Becoming a Partner (http://www.scaledagile.com/become-a-partner/)

Partner Directory (http://www.scaledagile.com/listingcategory/directory/)

Partner Event Calendar (http://www.scaledagile.com/event-list/)

GET SOCIAL

Twitter (https://twitter.com/ScaledAgile)

Linkedin (https://www.linkedin.com/grps/Scaled-Agile-Framework-4189072?)

YouTube (https://www.youtube.com/user/scaledagile)

SlideShare (http://www.slideshare.net/ScaledAgile)

RECENT POSTS

Updated SAFe Scrum Master and SAFe Advanced Scrum Master courses now available! (/blog/updated-safe-scrum-master-and-safe-advanced-scrum-master-courses-now-available/)

Sep, 19th 2017

SAFe from the Trenches at Agile Australia (/blog/safe-from-the-trenches-at-agile-australia/)

Sep, 19th 2017

Updated SAFe Release Train Engineer and SAFe Product Owner/Product Manager courses now available! (/blog/updated-safe-release-train-engineer-and-safe-product-ownerproduct-manager-courses-now-available/)

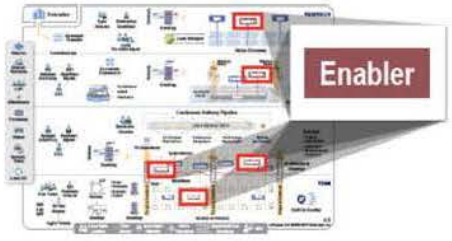
Aug, 31th 2017

SCALED AGILE, INC

CONTACT US

5480 Valmont Rd., Suite 100
Boulder, CO 80301 USA

Email: support@scaledagile.com (mailto:support@scaledagile.com)



(/)

> *Luck is what happens when preparation meets opportunity.*
>
> —Seneca

Enablers

Enablers promote the activities needed to extend the architectural runway to support future business functionality. These include exploration, infrastructure, compliance, and architecture development. They are captured in the various backlogs and occur at all levels of the framework.

Enablers (reflected in red on the Big Picture) exist on all four levels of SAFe: *Enabler Epics* (/epic/) at the Portfolio Level (/portfolio-level/), *Enabler Capabilities* (/features-and-capabilities/) at the Large Solution Level (/large-solution-level/), *Enabler Features* (/features-and-capabilities/) at the Program Level (/program-level/), and *Enabler Stories* (/story/) at the Team Level (/team-level/). They can be used for any activities that support upcoming business requirements, but generally fall into one of four categories:

Exploration – Exploration enablers are used for research, prototyping, and other activities needed to develop an understanding of Customer (/customer/) needs, to explore prospective Solutions (/solution/), and to evaluate alternatives.

Architecture – Architectural enablers are used to build the Architectural Runway, which allows smoother and faster development.

Infrastructure – Infrastructure enablers build, enhance and automate the development, testing, and deployment environments. They facilitate faster development, higher-quality testing, and a faster Continuous Delivery Pipeline.

Compliance – Compliance enablers are used to schedule and manage specific compliance activities, including Verification and Validation (V&V), documentation and signoffs, and regulatory submissions and approvals.

Details

Enablers are the tools that capture information and bring visibility to all the work necessary to support efficient development and delivery of future business requirements. Primarily, they're used for exploration, architectural Solution (/solution/) evolution, compliance (/compliance/) activities, and to enhance development and testing environments (see later sections of this article). Since they reflect the real work (and sometimes plenty of it) they cannot remain invisible. Rather, they're treated like all other value-added development activities—subject to estimating, visibility and tracking, Work In Process (WIP) limits, feedback, and presentation of results.

Types of Enablers

Enablers exist at all levels of the Framework (and are indicated on the Big Picture in red shading). They assume the attributes of the work they are associated with, as follows:

Enabler epics are written using the recommended Epic hypothesis statement format. They tend to cut across Value Streams (/value-streams/) and Program Increments (PIs) (/program-increment/). To support their implementation, they must include a

(/value-streams/) and Program Increments (PIs) (/program-increment/). To support their implementation, they must include a Lean business case and are identified and tracked through the Portfolio Kanban (/portfolio-kanban/) system.

Enabler capabilities and features occur at the Large Solution (/large-solution-level/) and Program Levels (/program-level/) to capture work of that type. Since these enablers are a type of Feature or Capability (/features-and-capabilities/), they share the same attributes, including a statement of benefit hypothesis and acceptance criteria. They also must be structured to fit within a single PI.

Enabler stories, like any Story (/story/), must fit in Iterations (/iterations/). Although they may not require the user voice format, their acceptance criteria clarifies the requirements and support testing.

Enablers are written and prioritized to follow the same rules as their respective epics, capabilities, features, and stories.

Creating and Managing Enablers

Most enablers are created when business initiatives, making their way through the different Kanban systems, require the additional support and elaboration shown below:

- *Exploration enablers* to validate the need or solution
- *Architectural enablers* to pave the runway
- *Infrastructure enablers* to develop, test, and integrate the initiatives
- *Compliance enablers* to prepare compliance activities

Enablers are often created by architects or by systems engineering at the various levels. They might be Enterprise Architects (/enterprise-architect/) at the Portfolio Level (/portfolio-level/), or Solution and System Architects/Engineering (/system-and-solution-architect-engineering/) at the Large Solution (/large-solution-level/) and Program Levels (/program-level/). These architects steer their enablers through the Kanban systems, providing the guidance to analyze them, and the information to estimate and implement them.

To improve the existing solution, some enablers will emerge locally from the needs of the Agile Teams (/agile-teams/), Agile Release Trains (ARTs) (/agile-release-train/), or Solution Trains (/solution-train/), to ensure that enough emphasis is placed on furthering the solution and extending the Architectural Runway (/architectural-runway/). Those that make it through the Kanban systems will be subject to capacity allocation in the Program and Solution Backlogs (/program-and-solution-backlogs/). This can be applied for enabler work as a whole, or it can distinguish between different types of enablers.

Using Enablers

Exploration

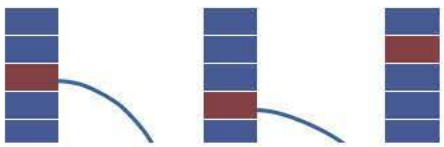
Applying *Enablers* for exploration provides a way for development teams to flesh out the details of requirements and design. The nature of Solution Intent (/solution-intent/) is that many requirements begin as variables. After all, at the beginning of development little is known about what the Customer (/customer/) needs or how to implement it. Customers themselves often don't understand exactly what they want. Only through iterative product development and demos do they begin to figure out what they actually need.

On the solution side, there are many technical possibilities for how to implement a business need. Those alternatives must be analyzed and are often evaluated through modeling, prototyping, or even concurrent development of multiple solution options (also known as set-based engineering).

Architecture

The architectural runway is one of the constructs SAFe uses to implement the concepts behind Agile Architecture (/agile-architecture/). The runway is the basis for developing business initiatives more quickly, on appropriate technical foundations. But the runway is constantly consumed by business epics, features, capabilities, and stories; so it must be maintained. Enablers are the backlog items used to maintain and extend the runway.

Some architectural enablers fix existing problems with the solution—for example, the need to enhance performance. These enablers start out in the backlog, but after implementation they may become Nonfunctional Requirements (/nonfunctional-requirements/) (NFRs). In fact, many NFRs come into existence as a result of architectural enablers, and tend to build over time, as can be seen in Figure 1.



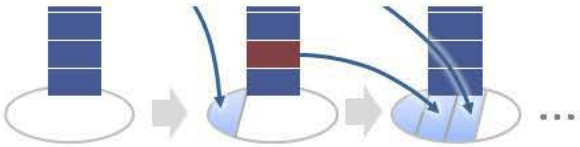


Figure 1. Many nonfunctional requirements appear over time as a result of enablers

Infrastructure

Agile development is built on frequent integration. Agile Teams integrate their work with other teams on the ART at the System Demos (/system-demo/) in every iteration. The trains integrate every Program Increment (PI) for the Solution Demo (/solution-demo/). Many Enterprises (/enterprise/) implement Continuous Integration (/continuous-integration/) and Continuous Deployment to ensure that the solution is always running. This reduces the risk at the integration points, and permits deployment and early release value.

To support such frequent or continuous integration and testing, there's a need to develop infrastructure at the Team Level (/team-level/), program level, and large solution level. Agile Teams, working with the System Team (/system-team/), are responsible for building and maintenance. Infrastructure enablers are used as backlog items to advance this work and continuously enhance it, both to support new scenarios and to enhance the agility of the enterprise.

Compliance

By incrementally building the necessary artifacts in the solution content over a series of PIs, SAFe supports continuous Verification and Validation (/compliance/). *Verification* activities are implemented as part of the flow (e.g., backlog items or Definition of Done (DoD)). While the artifacts will satisfy the objective evidence needed at the *end* of development, they are created iteratively throughout the lifecycle. *Validation* occurs when Product Owners, customers, and end users participate in ART planning and system demos, validating fitness for purpose.

For example, consider that regulations require design reviews, and that all actions need to be recorded and resolved. The design review enabler backlog item offers evidence of the review, and its DoD ensures that actions are recorded and resolved according to the Lean QMS (/compliance/). If needed, the actions themselves could be tracked as enabler stories. Regulations may also require that all changes be reviewed, which is addressed by a compulsory 'peer review' DoD for all stories.

Implement Architectural Enablers Incrementally

The size and demands of architectural enabler work can make it overwhelming. So it's important to remember that it needs to be broken down into small enabler stories that can fit in iterations. This can be difficult, however, as architectural and infrastructure changes can potentially stop the existing system from working until the new architecture/infrastructure is in place. When planning enabler work, make sure to organize it so that the system can run for most of the time on the old architecture or infrastructure. That way, teams can continue to work, integrate, demo, and even release while the enabler work is happening.

As described in System and Solution Architect/Engineering (/system-and-solution-architect-engineering/) and in footnote [1], there are three options:

- Case A: The enabler is big, but there is an incremental approach to implementation. The system always runs.
- Case B: The enabler is big, but it can't be implemented entirely incrementally. The system will need to take an occasional break.
- Case C: The enabler is *really* big, and it can't be implemented incrementally. The system runs when needed. In other words, do no harm.

Examples of incremental patterns are also described in footnote [2], Chapter 2, in which the legacy subsystems are gradually "strangled" over time, using proven patterns such as asset capture or event interception.

By creating the technology platforms that deliver business functionality, enablers drive better economics. But innovative product development cannot occur without risk-taking. Therefore, initial technology-related decisions cannot always be correct. Which is why the Agile enterprise must be prepared to reverse course on occasion. The Ignore-Sunk Costs-principle of product development flow (see footnote [3], principle E17) provides essential guidance: Do not consider money already spent. Incremental implementation helps, as corrective action can be taken before the investment grows too large.

Plan Cross-ART and Cross-Value Stream Enablers

Enabler epics and enabler capabilities can cut across multiple value streams or ARTs. During the analysis phase of the

Kanban system, one of the important decisions to make is whether to implement the enabler in all Solution Trains/ARTs at the same time, or to do so incrementally. This is a trade-off between the risk reduction of implementing one solution or system at a time vs. the Cost of Delay (CoD) caused by not having the full enabler, as Figure 2 illustrates.

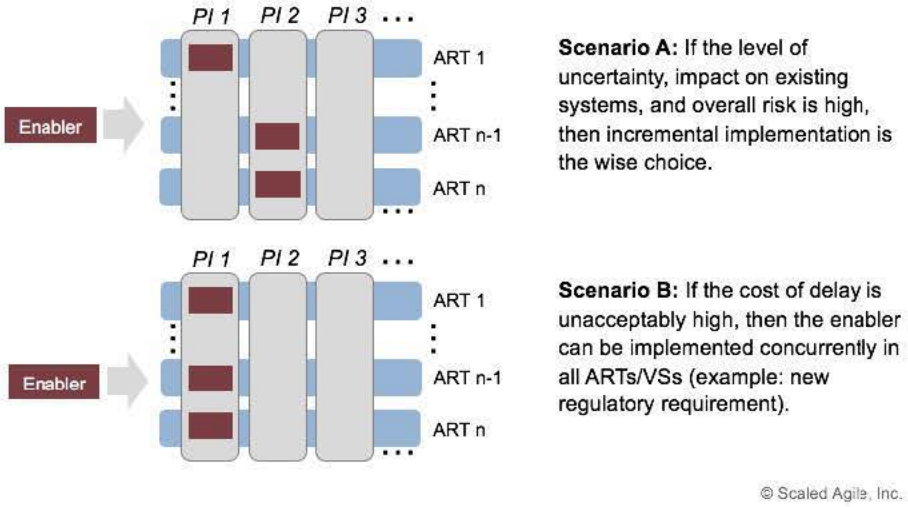


Figure 2. Two scenarios for implementing large enablers spanning multiple ARTs or Value Streams

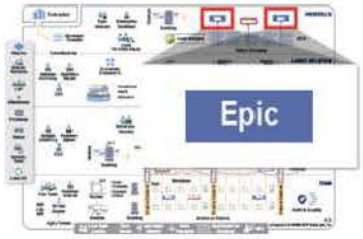
Learn More

[1] Leffingwell, Dean. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison-Wesley, 2011.

[2] Fowler, Martin. Strangler Application. http://martinfowler.com/bliki/StranglerApplication.html (http://martinfowler.com/bliki/StranglerApplication.html).

[3] Reinertsen, Donald. *The Principles of Product Development Flow: Second Generation Lean Product Development*. Celeritas Publishing, 2009.

Last update: 8 June, 2017



(/)

> *What if we found ourselves building something that nobody wanted? In that case what did it matter if we did it on time and on budget?*
>
> —Eric Ries

Epics

An *Epic* is a container for a Solution development initiative large enough to require analysis, definition of a Minimum Viable Product (MVP), and financial approval prior to implementation. Implementation occurs over multiple Program Increments (PIs) and follows the Lean Build-Measure-Learn startup cycle.

Epics are integral to Lean Portfolio Management (/lean-portfolio-management/) (LPM) and Lean Budgeting (/lean-budgets/) models. They require an Epic Owner (/epic-owner/) and Lean business case. Epics typically do not require a traditional scope completion end state. Instead, work continues until the optimum economic benefit is achieved. (See Continuous Delivery Pipeline (/continuous-delivery-pipeline/).)

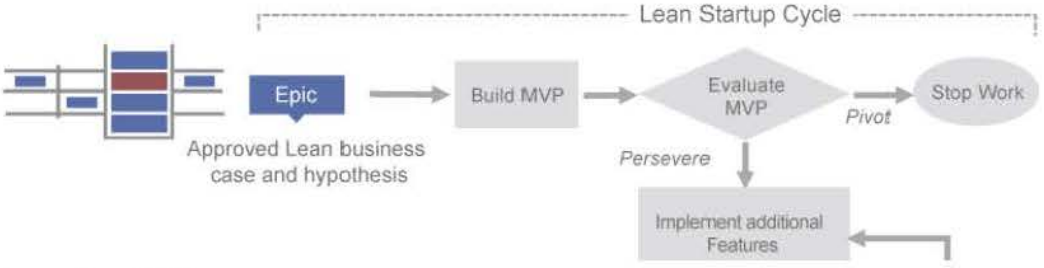
There are two types of epics: business epics and enabler epics, each of which may occur at the Portfolio (/portfolio-level/), Large Solution (/large-solution-level/), and Program Levels (/program-level/). This article primarily describes the definition, approval, and implementation of portfolio epics. Solution and program epics follow a similar pattern.

Details

Epics are the containers that capture and manage the largest initiatives that occur within a portfolio. They, and the Value Streams they affect, are the primary concern of the portfolio level. Business epics directly deliver business value; enabler epics are used to advance the Architectural Runway (/architectural-runway/) to support upcoming business epics.

Fostering Innovation with the Lean Startup Cycle and Lean Budgeting

Based in part on the emergence of Agile methods, the Lean Startup [1] strategy recommends a highly iterative Build-Measure-Learn cycle for product innovation and strategic investments. Applying this model to epics provides the economic and strategic advantages of a Lean startup—managing investment and risk incrementally—while leveraging the flow and visibility constructs that SAFe provides. (See Figure 1.)(Note: for further discussion of this figure, see Continuous Delivery Pipeline (/continuous-delivery-pipeline/).)



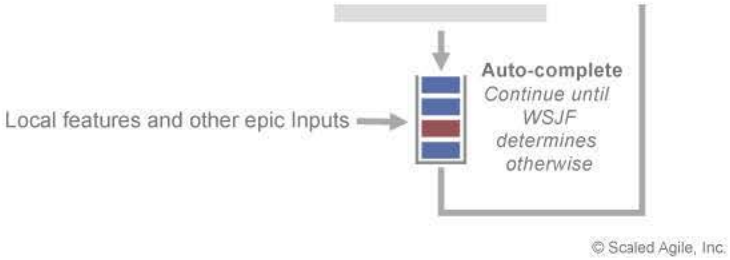


Figure 1. Epics in the Lean Startup Cycle

In addition, the empowerment and decentralized decision-making of Lean budgeting (/lean-budgets/) depend on certain checks and balances. Even in the context of an approved value stream budget, epics still require visibility and approval of an MVP estimate prior to implementation. Thereafter, further investment in the epic is controlled locally via ongoing Feature prioritization of the Program Backlog (/program-and-solution-backlogs/).

Overview of the Portfolio Kanban

Portfolio epics are made visible, developed, and managed through the Portfolio Kanban (/portfolio-kanban/), where they proceed through various states of maturity until they're rejected or approved. Understanding the Kanban system is fundamental to understanding of portfolio epics. The states of the Portfolio Kanban are summarized below.

- **Funnel** – The capture state
- **Review** – Preliminary estimates of opportunity, effort, and cost of delay
- **Analysis** – Establish viability, outcomes, MVP, development and deployment impact, Lean business case, go/no-go decision
- **Portfolio Backlog** – Epics with go approval move to the portfolio backlog (/portfolio-backlog/) until implementation capacity is available
- **Implementation** – Implementation begins when capacity becomes available

Defining Epics

Reasoning about a potential epic must be based on a definition and intent that stakeholders can agree to. Figure 2 provides an epic hypothesis statement template that can be used to capture, organize, and communicate key information about an epic.

Epic Hypothesis Statement	
For	<customers>
who	<do something>
the	<solution>
is a	<something – the "how">
that	<provides this value>
Unlike	<competitor, current solution, or non-existing solution>
our solution	<does something better – the "why">
Outcomes hypothesis:	• •
Leading indicators:	• (early innovation accounting measures) •
NFRs:	• •

Figure 2. Epic hypothesis statement

Analyzing and Approving Epics

Epics must be analyzed before being committed to implementation. Epic Owners (/epic-owner/) take responsibility for this important task, while Enterprise Architects (/enterprise-architect/) shepherd the enabler epics that support the technical considerations for business epics. The worthiest in the funnel pass to analysis when the queue has space. There, effort and economic impact are better defined, Weighted Shortest Job First (WSJF) prioritization is refined, and cost estimates are established.

The result of the analysis phase is a Lean business case. A Lean business case sample format is provided in Figure 3 below.

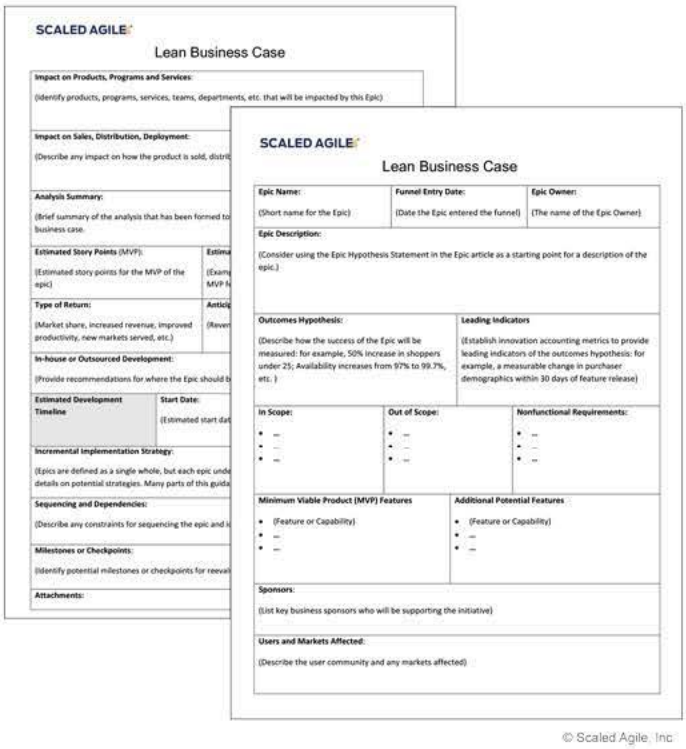


Figure 3. Epic Lean business case

The appropriate authorities then employ the Lean business case to make a go/no-go decision for the epic.

You can download the epic Lean business case template here.

Download Lean Business Case (http://www.scaledagileframework.com/?ddownload=35395)

Implementing Epics

Once approved, portfolio epics stay in the portfolio backlog until implementation capacity becomes available. The Epic Owner or Enterprise Architect has the responsibility to work with the Product Managers and System Architect/Engineering to define the MVP. Further, they split the epics into large solution and program-level epics, or directly into capability or feature backlog items during implementation.

Splitting Epics

Implementing this way means that an epic must be split into smaller pieces that represent incremental value. This is the art and skill of building large systems incrementally [3]. Table 1 suggests nine methods for splitting epics, along with examples for each:

1. Solution / Subsystem / Component – Epics often affect multiple solutions (/solution/), subsystems, or large components. In such cases, splitting by these aspects can be an effective implementation technique.	
Multiple user profiles	*… Multiple profiles in the opt-out website* *… Multiple profiles in the admin system*
2. Success Criteria – The epic success criteria often provide hints as to how to incrementally achieve the anticipated business value.	
Implement new artifact in search results: Locations success criteria: a) Locations should provide additional filtering method when other disambiguation methods aren't useful b) Provide detailed location of a person	*… Provide state information in the search results* (criteria [a] is partially satisfied, as states alone already provide some good filtering capability) *… Implement compound location:* State and city (entire success criteria is satisfied)
3. Major Effort First – Sometimes an epic can be split into several parts, where most of the effort will go toward implementing the first one.	
Implement single sign-on across all products in the suite	*… Install PINGID protocol server and test with mock identity provider* *…Implement SSO management capability in our simplest product* *… Implement SSO in our most complex product* *… Proliferate as quickly as backlog capacity allows*

4. Simple/Complex – Capture that simplest version as its own epic, and then add additional program epics for all the variations and complexities.	
5. Variations in Data – Data variations and data sources are another source of scope, complexity, and implementation management.	
Internationalize all end-user facing screens	*... in Spanish* *... in Japanese* *... prioritize the rest by then-current market share*
6. Market Segment / Customer / Class of User – Segmenting the market or customer base is another way to split an epic. Do the one that has higher business impact first.	
Implement opt-in functionality	*... For current partners* *... For all major marketers*
7. Defer Solution Qualities (NFRs) – Sometimes the initial implementation isn't all that hard, and the major part of the effort is in making it fast—or reliable or more precise or more scalable—so it may be feasible to achieve the solution qualities (Nonfunctional Requirements (/nonfunctional-requirements/)) incrementally.	
8. Risk Reduction | Opportunity Enablement – Given their scope, epics can be inherently risky; use risk analysis and do the riskiest parts first.	
Implement filtering search results by complex user-defined expression	*... Implement negative filtering* *... Implement complex filtering expressions with all logical operations*
9. Use Case Scenarios – Use cases [1] can be used in Agile to capture complex user-to-solution or solution-to-solution interaction; split according to specific *scenarios* or *user goals* of the use case.	
Transitive people search functionality	(Goal 1 ~) *Find connection to a person* (Goal 2 ~) *Find connection to a company* (Goal 3 ~) *Distinguish strong and weak connections*

Table 1. Methods for splitting epics

Solution and Program Epics

The above discussion describes the largest kind of epics, portfolio epics. These are typically cross-cutting and affect multiple value streams. Many generate solution epics and, correspondingly, program epics. In addition, many such epic-level initiatives arise at the solution or program levels. While largely a local concern, their impact on financial, human, and other resources is large enough to warrant a Lean business case, discussion, and financial approval from Lean Portfolio Management. That's what makes an epic an epic.

Methods for managing epics at these levels are discussed in the Program and Solution Kanban (/program-and-solution-kanbans/) article.

Learn More

[1] Ries, Eric. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Random House, Inc. Kindle Edition, 2011.

[2] Leffingwell, Dean. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison-Wesley, 2011.

[3] Yakyma, Alex. Implementation Strategies for Business Epics (/Implementation Strategies for Business Epics).

Last update: 16 September, 2017

TRAINING

Course Calendar (http://www.scaledagileacademy.com/events/event_list.asp)

About Certification (http://www.scaledagileacademy.com/?page=WhichCertification)

Become a Trainer (http://www.scaledagileacademy.com/?page=becomeatrainer)

PERMISSIONS

Permission FAQ (http://www.scaledagile.com/permission-faq/)

Permissions Form (http://www.scaledagile.com/permissions-form/)

Usage and Permissions (/usage-and-permissions/)

PARTNER

Becoming a Partner (http://www.scaledagile.com/become-a-partner/)

Partner Directory (http://www.scaledagile.com/listingcategory/directory/)

Partner Event Calendar (http://www.scaledagile.com/event-list/)

GET SOCIAL

Twitter (https://twitter.com/ScaledAgile)

Linkedin (https://www.linkedin.com/grps/Scaled-Agile-Framework-4189072?)

YouTube (https://www.youtube.com/user/scaledagile)

SlideShare (http://www.slideshare.net/ScaledAgile)

RECENT POSTS

Updated SAFe Scrum Master and SAFe Advanced Scrum Master courses now available! (/blog/updated-safe-scrum-master-and-safe-advanced-scrum-master-courses-now-available/)

Sep, 19th 2017

SAFe from the Trenches at Agile Australia (/blog/safe-from-the-trenches-at-agile-australia/)

Sep, 19th 2017

Updated SAFe Release Train Engineer and SAFe Product Owner/Product Manager courses now available! (/blog/updated-safe-release-train-engineer-and-safe-product-ownerproduct-manager-courses-now-available/)

Aug, 31th 2017

SCALED AGILE, INC

CONTACT US

5480 Valmont Rd., Suite 100
Boulder, CO 80301 USA

Email: support@scaledagile.com (mailto:support@scaledagile.com)

Phone: +1.303.554-4367

BUSINESS HOURS

Weekdays: 9am to 5pm
Weekends: CLOSED

SAFe® for Lean Enterprises

What's New in SAFe 4.5?

We are delighted to announce the release of SAFe 4.5, the result of research from hundreds of implementations, customer and community feedback, and advances made in the knowledge areas SAFe is built on. As you learn more, we believe you'll find SAFe 4.5 to be leaner, more Agile, and more supportive of faster innovation and learning than any of its predecessors. Moreover, SAFe 4.5 helps enterprises get better, business results, faster, more consistently and reliably.

SAFe 4.5 can be configured to match your organization's needs. This new version allows companies to:

- *Test ideas more quickly* using the Lean Startup Cycle and Lean User Experience (Lean UX)
- *Deliver much faster* with Scalable DevOps and the Continuous Delivery Pipeline.
- *Simplify governance and improve portfolio performance* with Lean Portfolio Management (LPM) and Lean Budgets.

But we all know that organizational change is hard. It requires adopting new behaviors, leadership styles, practices, and culture. SAFe accelerates Lean-Agile transformation with the new *Implementation Roadmap*, which guides enterprises every step of the way. And the SAFe Journey is supported by a worldwide network of Scaled Agile Partners (http://www.scaledagile.com/partners/) and *SAFe Program Consultants (SPCs)*.

SAFe 4.5 maximizes the speed of product or service delivery, from initial idea to release, and from customer feedback to enhancements, providing a 360-degree build-measure-learn feedback cycle.

The SAFe 4.5 release focuses on the following key areas of improvements:

1. Essential SAFe and Configurability
2. Innovation with Lean Startup and Lean UX
3. Scalable DevOps & Continuous Delivery
4. Implementation Roadmap
5. Other Important Stuff

What's New in SAFe 4.0 video on the SAFe Community Platform (https://community.scaledagile.com/CustomCommunityLogin) that will help you learn about the new features, and upgrade your certification so that you can support enterprises seeking SAFe 4.5 implementation.

Backwards Compatible with SAFe 4.0

SAFe 4.5 is backwards compatible with SAFe 4.0 (http://v4.scaledagileframework.com/), as shown in Figure 1 below. This means that organizations can adopt the new features in SAFe 4.5 at their own pace.

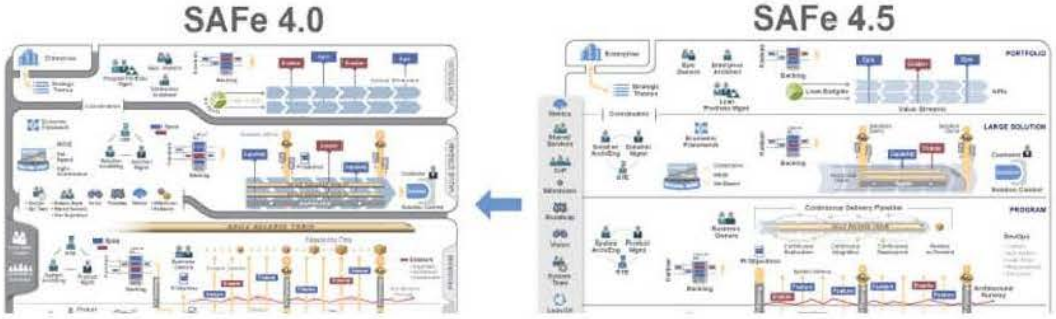
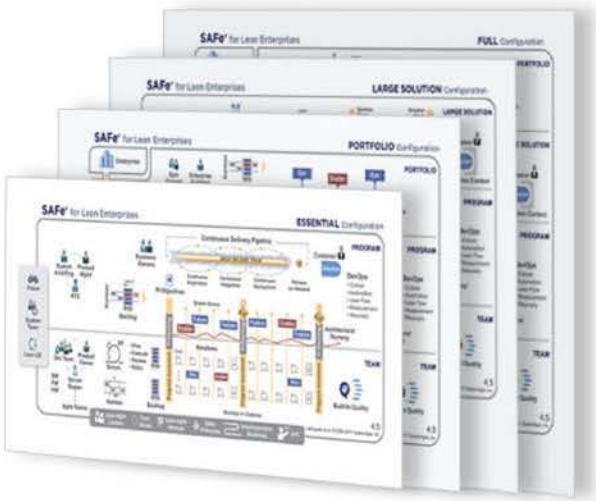


Figure 1. SAFe 4.5 is backwards compatible with SAFe 4.0

Essential SAFe and Configurability

SAFe's new configurable framework provides just enough guidance to meet the needs of your product or service. ***Essential SAFe*** helps companies sprint out of the gate quickly and start simply. As company needs grow, SAFe scales to meet those challenges.



(http://www.scaledagileframework.com/posters/)

DOWNLOAD POSTERS
(HTTP://WWW.SCALEDAGILEFRAMEWORK.COM/POSTERS/)

There are four configurations of SAFe that provide a more configurable and scalable approach:

1. Full SAFe
2. Portfolio SAFe
3. Large Solution SAFe
4. Essential SAFe

Using the menu shown in Figure 2, you can choose the SAFe configuration that's right for your needs. Whenever you return to the home page, SAFe ***remembers your choice***. Each configuration is described below.

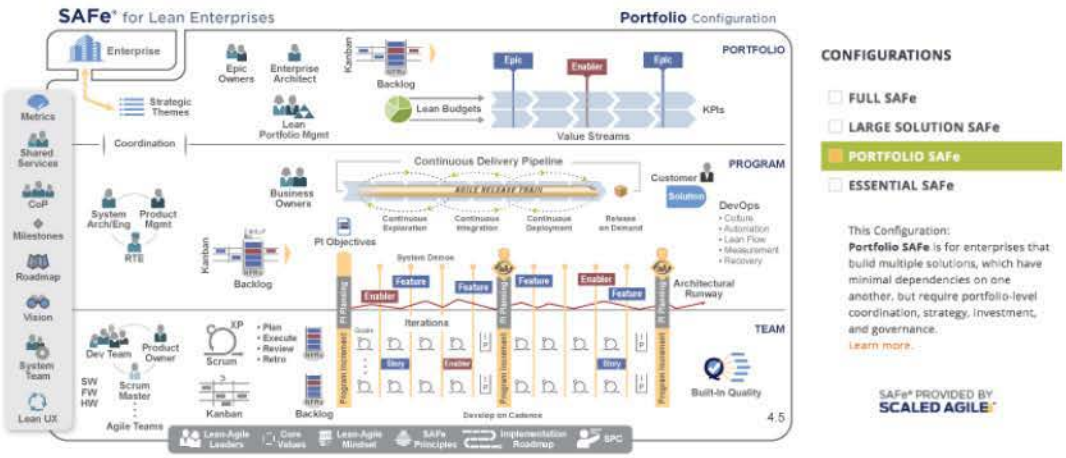


Figure 2. Portfolio SAFe

Essential SAFe

Figure 3 illustrates *Essential SAFe*, the most basic configuration. The starting point for implementing SAFe, it describes the most critical elements needed to realize the majority of the framework's benefits.

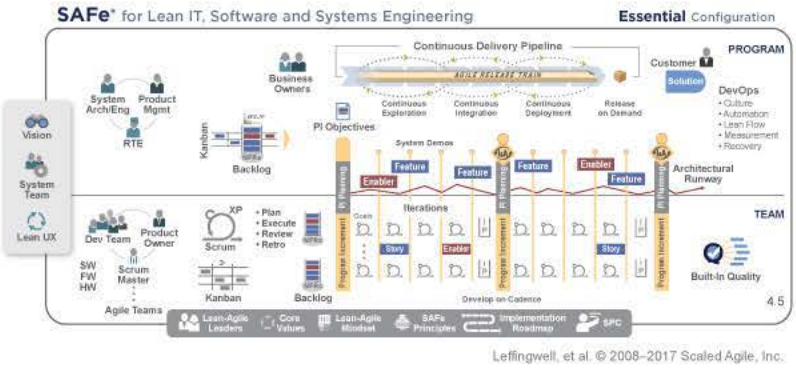


Figure 3. Essential SAFe

Portfolio SAFe

The *Portfolio SAFe* configuration (formerly 3-level SAFe) is for enterprises that build multiple, largely independent, solutions that also need to incorporate portfolio concerns. These may include strategy and investment funding, innovation across various value streams, and lean governance. It adds the Portfolio Level to Essential SAFe, as shown in Figure 4.

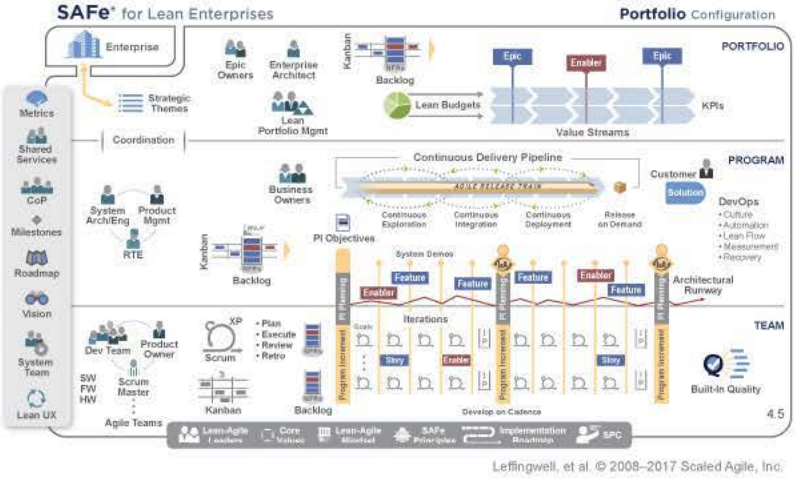
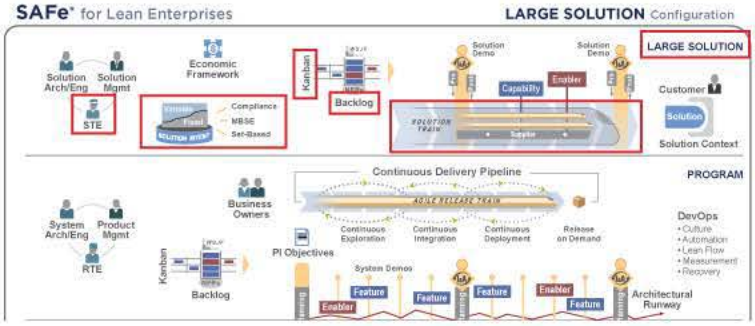


Figure 4. Portfolio SAFe

Large Solution SAFe

The *Large Solution SAFe* configuration is intended for enterprises that are building large and complex solutions that require the contribution of multiple Agile Release Trains and Suppliers, but do not require portfolio considerations. It consists of Essential SAFe and the Large Solution level, as illustrated in Figure 5.



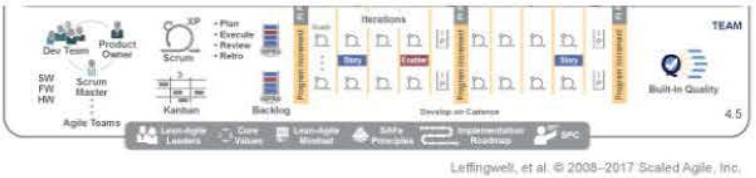


Figure 5. Large Solution SAFe

Along with other changes, note that the SAFe 4.0 Value Stream Level name has been changed to the '*Large Solution Level*'. This necessitated the terminology changes shown in Figure 6.

Previous Terminology	New Terminology
none	Solution Train
Value Stream Backlog	Solution backlog
Value Stream Engineer	Solution Train Engineer
Value Stream Epics	Solution Epics
Value Stream Kanban	Solution Kanban
Value Stream Level	Large Solution Level
Value Stream PI Objectives	Solution PI Objectives

Figure 6. Terminology changes resulting from renaming the Value Stream level

Build Big Systems with the New Solution Train

The Solution Train (Figure 7) describes the SAFe organizational construct used to build large and complex solutions that need to coordinate multiple Agile Release Trains (ARTs), as well as the contributions of Suppliers. Using a common Solution vision, backlog and roadmap—and an aligned Program Increment (PI) cadence—the Solution Train aligns ARTs and Suppliers around a shared business and technology mission.

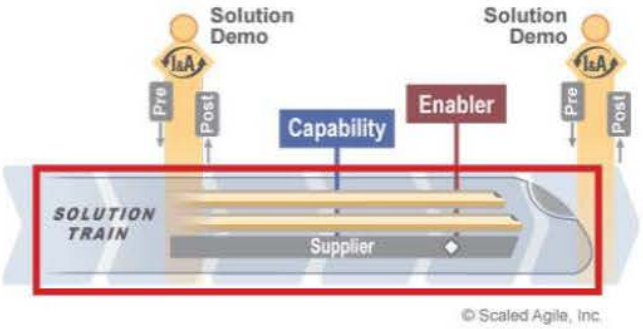


Figure 7. New Solution Train

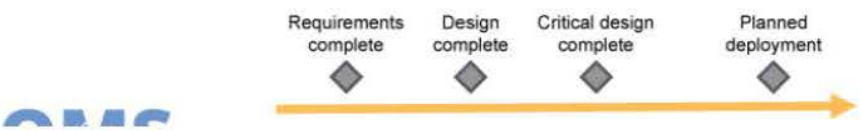
Solution Train Engineer (STE)

The Solution Train Engineer (STE) (/implementation-roadmap/), previously known as the Value Stream Engineer, is the train's servant leader. By identifying and resolving bottlenecks across the entire solution, the STE helps the Solution Train run smoothly.

Bringing Compliance into SAFe

Compliance (/compliance/) (Figure 8) is now part of *Solution Intent*. It describes how to achieve high quality results while meeting regulatory and industry requirements using Lean-Agile development.

Compliance *enablers* schedule and manage specific compliance activities, including Verification and Validation (V&V), documentation and signoffs, and regulatory submissions and approvals.



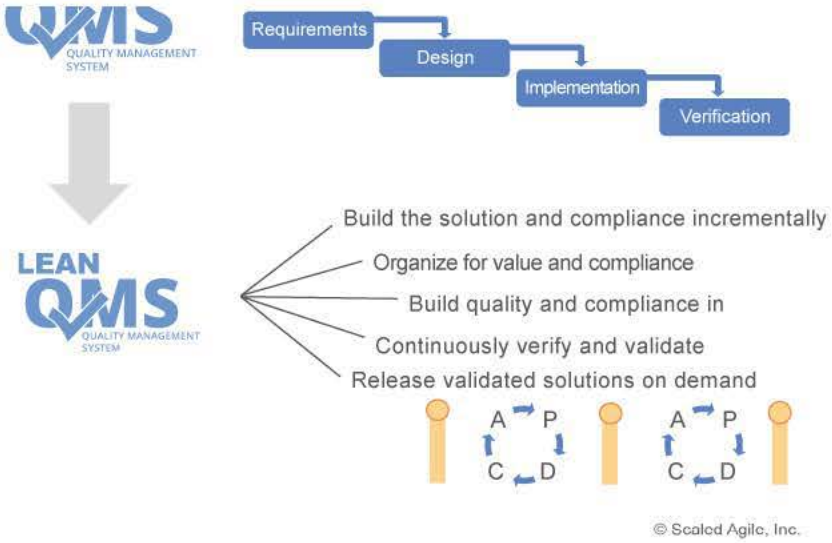


Figure 8. Introducing compliance into SAFe 4.5

Agile Architecture

Agile Architecture (/agile-architecture/) is still an element of Solution Intent. However, it has been moved off the big picture and into the guidance page. The Agile Architecture guidance can also be accessed from Built-In Quality, where it is discussed briefly.

Full SAFe

Full SAFe (Figure 9) is the most comprehensive configuration. It supports those enterprises building large, integrated solutions that typically require hundreds of people or more to develop and maintain.

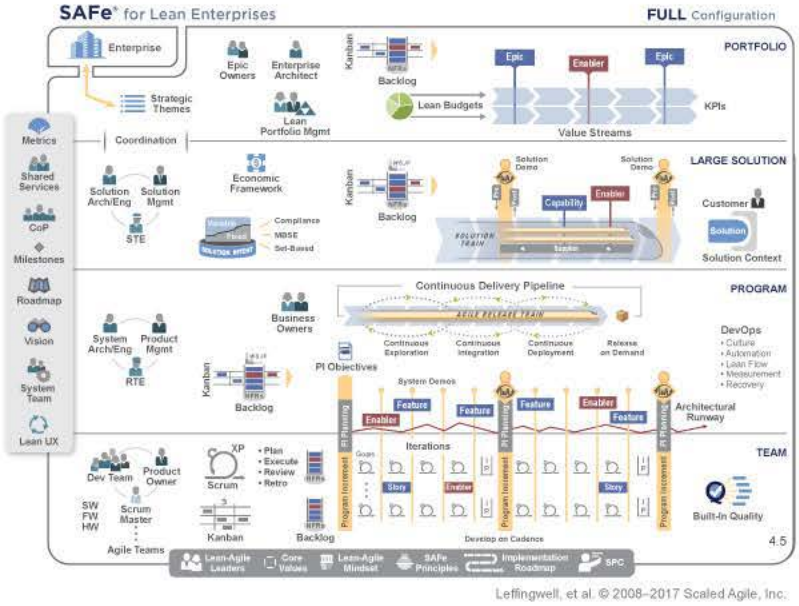


Figure 9. Full SAFe

Refine Configurations with the Spanning Palette

Changed from landscape to portrait orientation, the spanning palette was now truly spans all levels of the framework—team, program, large solution, and portfolio.

A key element of SAFe's flexibility and configurability, the spanning palette permits organizations to apply only the elements needed for each level. It dynamically adjusts depending upon the configuration chosen, as shown in Figure 10. Other changes include:

- Abbreviated Communities of Practice to 'CoP' due to the new spanning palette design
- Moved from UX to Lean UX and expanded on user experience development

- Moved from UX to Lean UX and expanded on user experience development
- Removed the Release Management Icon and Incorporated the content into Release on Demand (/release-on-demand/)

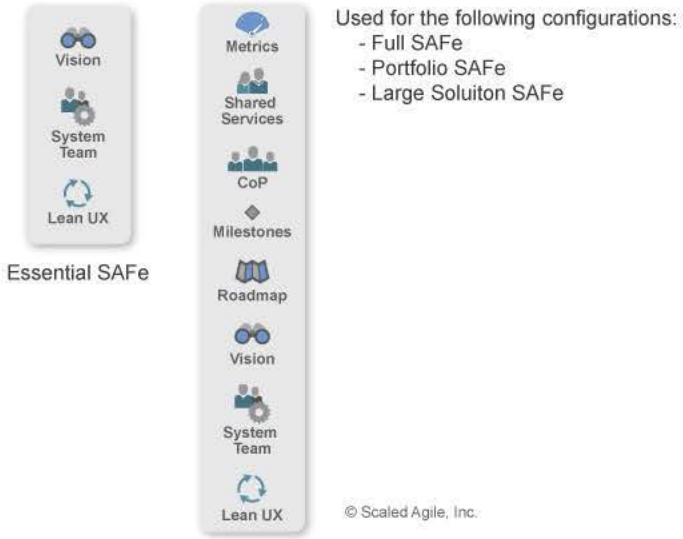


Figure 10. Spanning Palettes

Innovation with Lean Startup and Lean UX

The new content in Lean Startup Cycle, Lean UX, Lean Portfolio Management (LPM), and Lean Budgets helps you innovate rapidly and implement strategy more quickly to realize better business outcomes.

Lean Startup Cycle

The Lean Startup movement embraces the highly iterative "Hypothesize-Build-Measure-Learn" cycle, which fits quite naturally into SAFe. Specifically, this model can be applied to any Epic-level initiative, whether it arises at the Portfolio, Large Solution, or Program Level. No matter the source, the scope of an epic calls for a prudent and iterative approach to investment and implementation via a Minimum Viable Product (MVP), as reflected in Figure 11. Learn more about the Lean Startup cycle in the Epic (/epic) article.

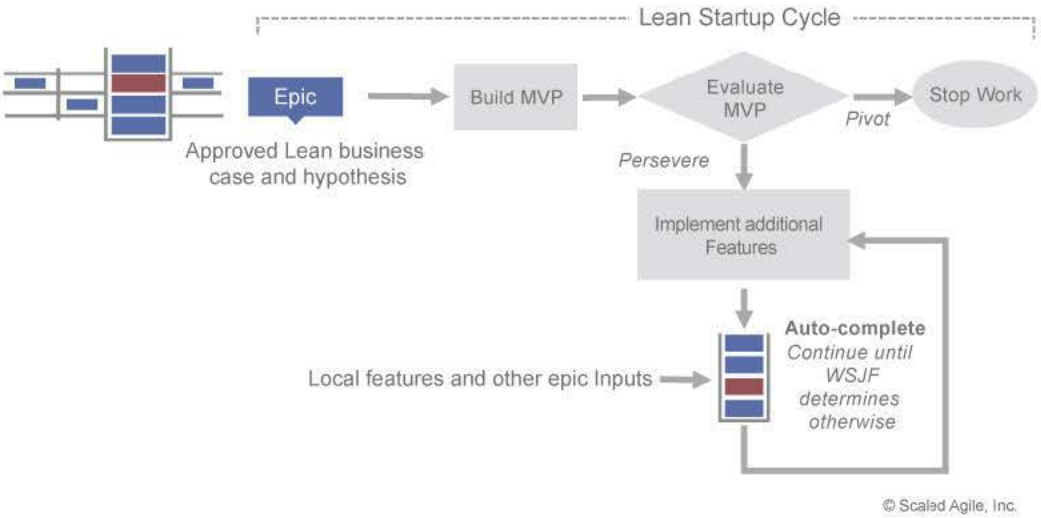


Figure 11. The Lean Startup Cycle

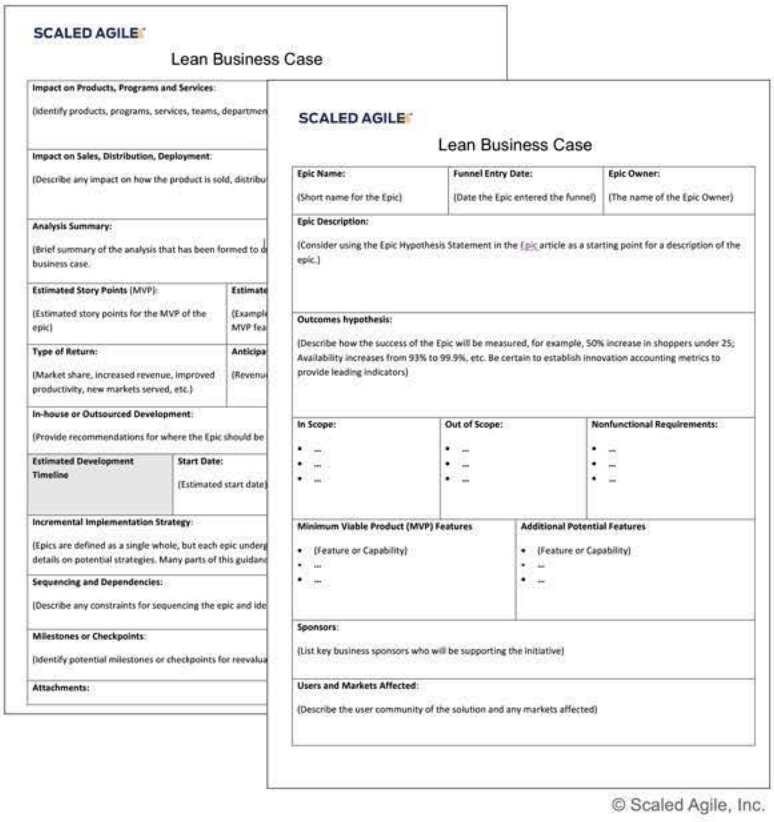
Epic Hypothesis and Lean Business Case

To foster the new Lean Startup approach in SAFe, the Epic Value Statement and Lightweight Business Case have been

refactored to become the *Epic Hypothesis Statement* (Figure 12) and *Lean Business Case* respectively (Figure 13). More than just name changes, they represent a whole new way of thinking and working. Learn more about the Epic Hypothesis Statement and Lean Business Case in the Epic (http://dev.scaledagileframework.com/epic/) article.

Epic Hypothesis Statement	
For	<customers>
who	<do something>
the	<solution>
is a	<something – the "how">
that	<provides this value>
Unlike	<competitor, current solution, or non-existing solution>
our solution	<does something better – the "why">
Outcomes hypothesis:	• •
Leading indicators:	• (early innovation accounting measures) •
NFRs:	• •

Figure 12. Epic Hypothesis Statement


SCALED AGILE

Lean Business Case

Impact on Products, Programs and Services:

(Identify products, programs, services, teams, departmen

Impact on Sales, Distribution, Deployment:

(Describe any impact on how the product is sold, distribu

Analysis Summary:

(Brief summary of the analysis that has been formed to d business case.

Estimated Story Points (MVP):	**Estimate**
(Estimated story points for the MVP of the epic)	(Exampl MVP fea
Type of Return:	**Anticipa**
(Market share, increased revenue, improved productivity, new markets served, etc.)	(Revenu

In-house or Outsourced Development:

(Provide recommendations for where the Epic should be

Estimated Development Timeline	**Start Date:** (Estimated start date)

Incremental Implementation Strategy:

(Epics are defined as a single whole, but each epic underg details on potential strategies. Many parts of this guidan

Sequencing and Dependencies:

(Describe any constraints for sequencing the epic and ide

Milestones or Checkpoints:

(Identify potential milestones or checkpoints for reevalua

Attachments:

SCALED AGILE

Lean Business Case

Epic Name:	**Funnel Entry Date:**	**Epic Owner:**
(Short name for the Epic)	(Date the Epic entered the funnel)	(The name of the Epic Owner)

Epic Description:

(Consider using the Epic Hypothesis Statement in the Epic article as a starting point for a description of the epic.)

Outcomes hypothesis:

(Describe how the success of the Epic will be measured, for example, 50% increase in shoppers under 25; Availability increases from 93% to 99.9%, etc. Be certain to establish innovation accounting metrics to provide leading indicators)

In Scope:	**Out of Scope:**	**Nonfunctional Requirements:**
• ... • ... • ...	• ... • ... • ...	• ... • ... • ...

Minimum Viable Product (MVP) Features	**Additional Potential Features**
• (Feature or Capability) • ... • ...	• (Feature or Capability) • ... • ...

Sponsors:

(List key business sponsors who will be supporting the initiative)

Users and Markets Affected:

(Describe the user community of the solution and any markets affected)

Figure 13. Lean Business Case

Lean User Experience

The Lean UX (http://dev.scaledagileframework.com/lean-ux/) process (Figure 14) starts with an outcome hypothesis: Agile teams and UX designers accept that the 'right answer' is actually unknowable in advance. Rather, they apply Agile methods to avoid Big Upfront Design (BUFD), focusing instead of creating a hypothesis about what business outcomes to expect from a new feature. They then implement and test the hypothesis incrementally. This results in faster feedback, which steers the solution toward success more effectively.


Feature

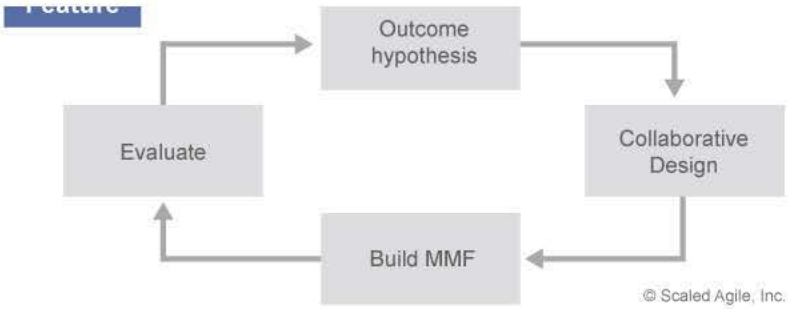


Figure 14. Lean UX

As a result of the hypothesis-driven approach of Lean UX, the Features and Benefits (FAB) matrix has been updated to describe features and capabilities with a ***Benefit Hypothesis,*** as illustrated in Figure 15.

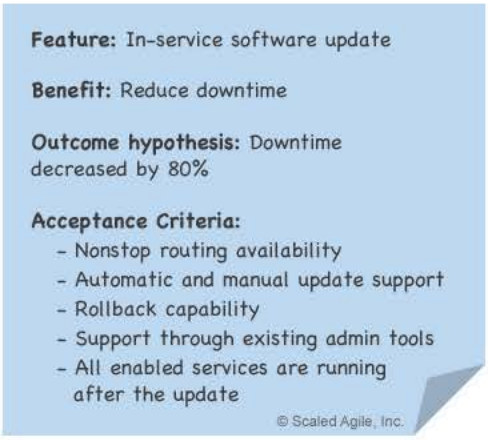


Figure 15. Features have a benefit hypothesis

Lead and Govern with Lean Portfolio Management

Formerly Program Portfolio Management (PPM), each SAFe portfolio has an LPM function that is responsible for strategy and investment funding, Agile program guidance, and Lean governance, as illustrated in Figure 16.

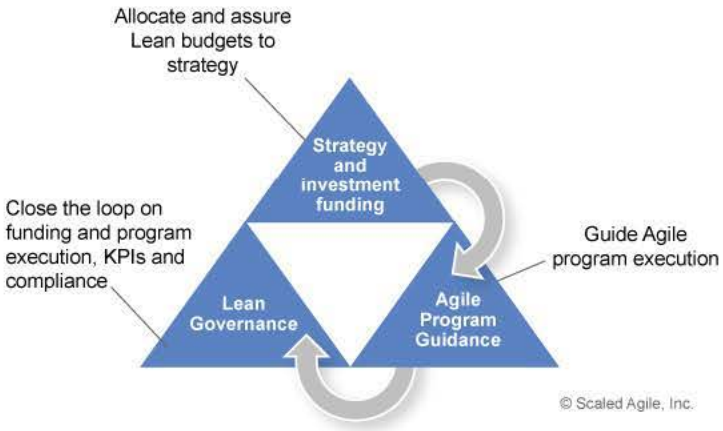


Figure 16. Lead and govern with Lean Portfolio Management

Scalable DevOps & the Continuous Delivery Pipeline

Together, scalable DevOps and the continuous delivery pipeline helps accelerate the build-measure-learn cycle that supports faster innovation and more frequent releases.

DevOps (/devops/) is a mindset, a culture, and a set of technical practices. It provides communication, integration, automation, and close cooperation among everyone needed to plan, develop, test, deploy, release, and maintain a Solution

automation, and close cooperation among everyone needed to plan, develop, test, deploy, release, and maintain a solution.

SAFe enterprises implement DevOps to break down silos and empower each Agile Team, ART and Solution Train to continuously deliver new features to end users. This is indeed achievable, as "high-performing IT organizations deploy 30x more frequently with 200x shorter lead times ... 60x fewer failures and recover 168x faster." [1]

DevOps consists of the following elements, shown in Figure 17 and described briefly below:

Figure 17. Realize flow with a CALMR approach to DevOps

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

Continuous Delivery Pipeline

The Continuous Delivery Pipeline (/continuous-delivery-pipeline/) doesn't operate in a strict linear sequence. Rather, it's a learning cycle that allows teams to establish a number of *hypotheses*, *build* and deliver against them, *measure* results and *learn* from that work, as Figure 18 illustrates.

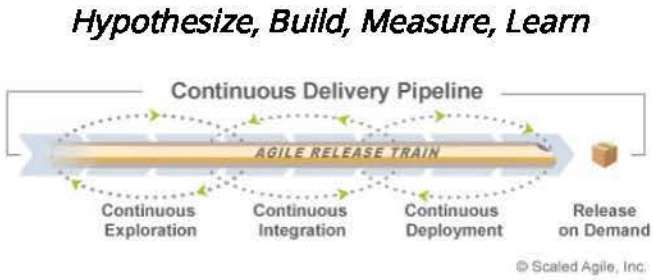


Figure 18. The SAFe Continuous Delivery Pipeline

- Continuous Exploration (/continuous-exploration/) – is the process of constantly exploring market and user needs, and defining a vision, roadmap, and set of features that address them.
- Continuous Integration (/continuous-integration/) – is the process of taking features from the Program Backlog and developing, testing, integrating, and validating them in a staging environment that prepares them for deployment and release.
- Continuous Deployment (/continuous-deployment/) – is the process that takes validated features from continuous integration and deploys them into the production environment, where they are tested and readied for release.
- Release on Demand (/release-on-demand/) – is the process by which deployed features are released to customers incrementally or immediately based on market demand.

The ART uses the Program Kanban (http://dev.scaledagileframework.com/program-and-solution-kanbans/) to facilitate the flow of features through the Continuous Delivery Pipeline. A typical program Kanban is illustrated in Figure 19. The policies applied

to each state, and some examples of WIP limits are highlighted. The example program Kanban *states* were updated to support DevOps, Continuous Delivery Pipeline, and releasing.

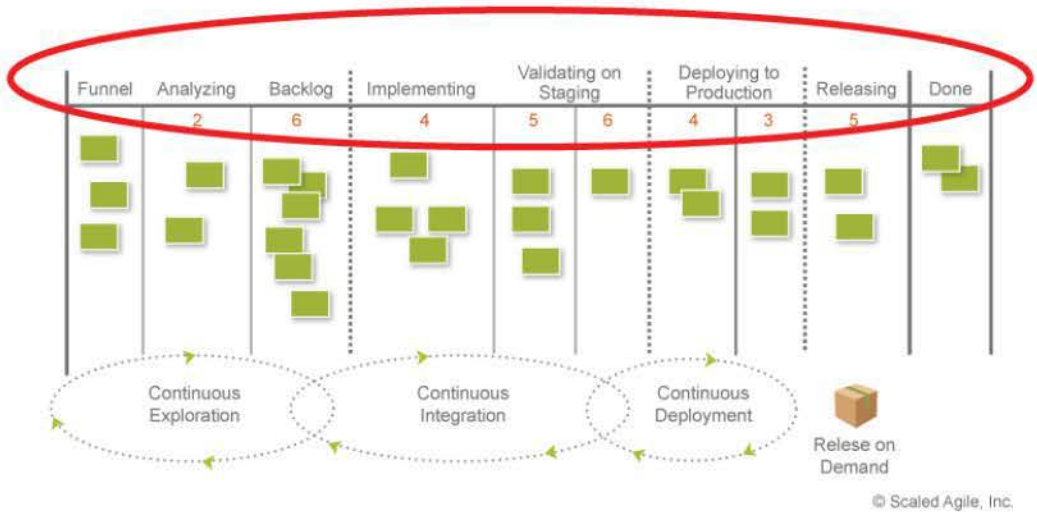


Figure 19. Visualize and manage full value flow

Implementation Roadmap

The SAFe Implementation Roadmap (/implementation-roadmap/) (Figure 20.) consists of a *clickable, interactive* graphic linked to a 12-article series that describes the major activities that have proven to be effective in successfully implementing SAFe.

The implementation roadmap article is accessible from both the foundation on the big picture and the implementing menu on the SAFe website. You can download a .pdf or .ppt version of the roadmap from this article.

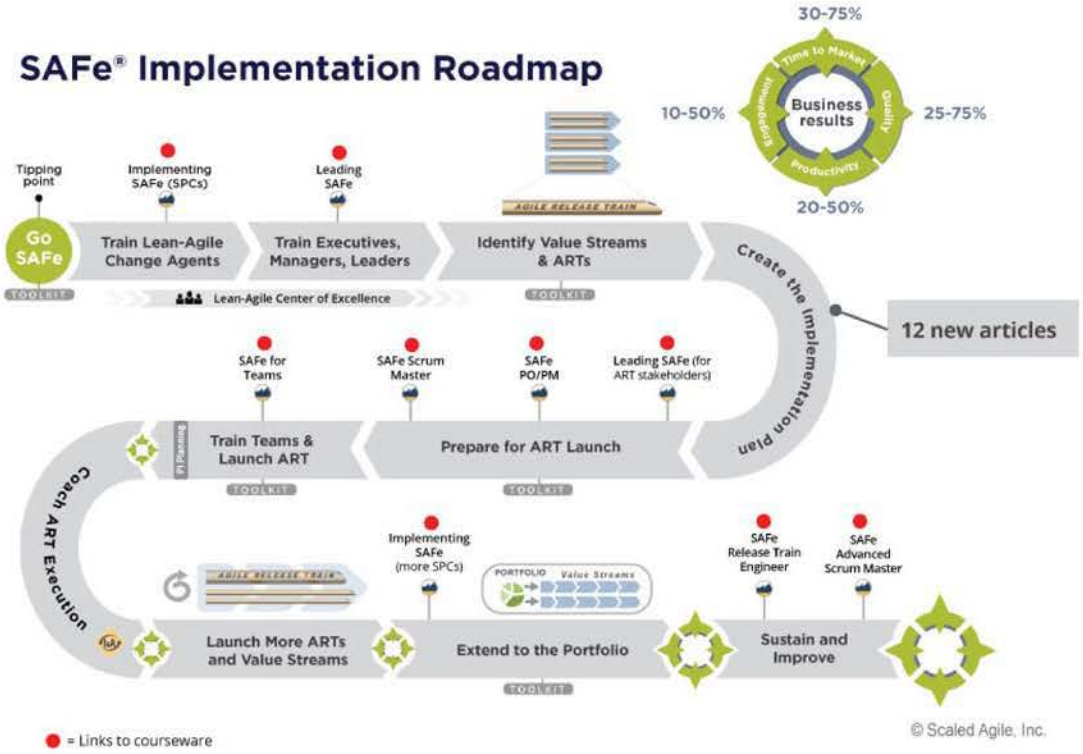


Figure 20. Implementation Roadmap

The critical role of the SAFe Program Consultant

SAFe Program Consultants (SPC) are change agents who combine their technical knowledge of SAFe with an intrinsic

motivation to improve their company's software and systems development processes. They play a critical role in successfully *Implementing SAFe*. SPC's are now represented as part of the SAFe Foundation and are described in the new SPC article, as illustrated in Figure 21.

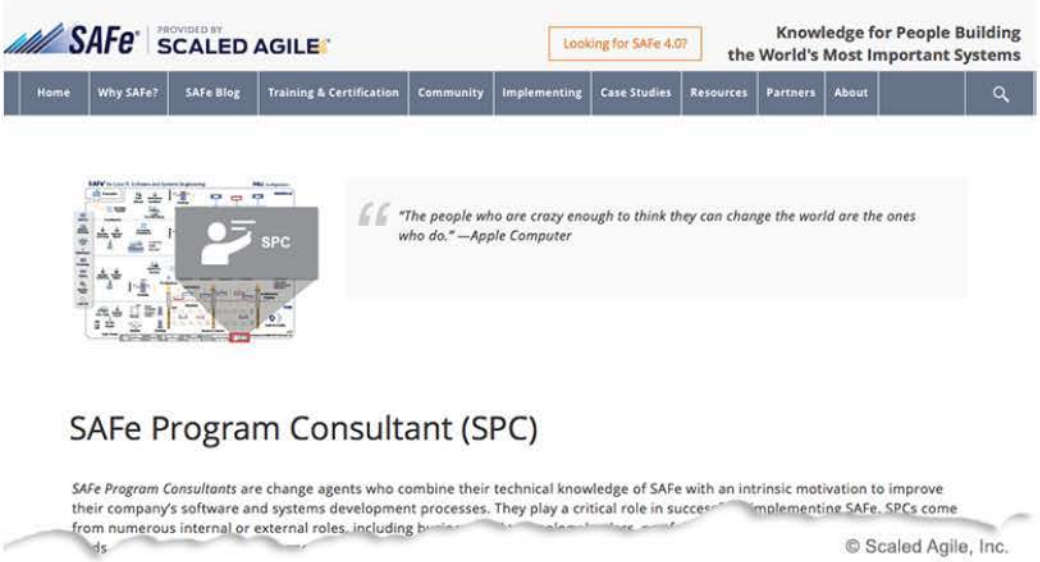


Figure 21. SAFe Program Consultant article excerpt

Other Important Stuff

What's New in SAFe Enablement Video

In addition to the new 4.5 content, all the 4.0 articles have been updated as well, so there is much more to learn, and benefit from. Anyone who is certified, will find a What's New in SAFe 4.0 video (https://community.scaledagile.com/CustomCommunityLogin) on the SAFe Community Platform that that highlights and explains the new features of the framework.

SAFe Version 4.0 Available Through June, 2018

We encourage you to adopt SAFe 4.5 to take advantage of the new features. But we also recognize that there are enterprises that will need the SAFe 4.0 version for some time to come. It will be supported until June, 2018, and remains available at v4.scaledagileframework.com (http://v4.scaledagileframework.com/).

Courseware Updates

Three courses from Scaled Agile's role-based curriculum have been already updated to reflect the new features of SAFe 4.5. They include: *Implementing SAFe, Leading SAFe, and SAFe for Teams.*

The remaining courses will be updated over the coming weeks. We hope this new version helps you and your enterprise achieve the benefits you deserve for working so hard at building the world's biggest, and best, software and systems.

More About SAFe's New Name

We think of SAFe as a big-tent framework that can accommodate any organization, regardless of size or complexity. As you know, SAFe is for business, *not just* for IT, systems, and software. It's part of a larger

solution development picture, and we want make sure that message is front and center for our internal and external business partners, and for the community at large. So we've updated the name to reflect this more inclusive theme. The Framework is now known as, ***SAFe for Lean Enterprises***.

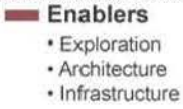
Improved Big Picture Look and Feel

We've improved the readability of the big picture by making hundreds of small changes. These include more white space, standardized font sizes and colors, and reduced visual clutter by moving less popular icons to guidance. For example:

- **CapEx & OpEx** – The icon was taken off the big picture and the article content was moved to the guidance page (/capex-and-opex/).

- **Release Management** – The icon was removed and the article content was incorporated into the Release on Demand (/release-on-demand/) article.

- **Enablers** – The enablers icon shown below was removed, but the types of enablers are still described in the Enablers (http://dev.scaledagileframework.com/enablers/) article. Removing this icon also increases the flexibility of SAFe to add new enablers (e.g. Compliance) without having an impact on the big picture.



- **Value Stream** and **Program Epics** – The icons were removed but the descriptions remain in the Epics. (/epics/)

Refactored Article Architecture

The format of articles was changed to improve consistency. Each article is consistently structured as follows:

- **Article title** – Shows the title of the article minus the word 'abstract' to reduce visual clutter
- **Glossary definition and overview** – Each article now includes the glossary definition first, followed by a brief introduction
- **Details** – Contains sub sections to describe the topic
- **Learn More** – Lists resources to learn more about the topic

Increased Alignment with the Scrum Guide

The following changes were made to improve alignment between SAFe and Scrum:

- Iteration Review (/iteration-review/) (Sprint Review) was added to the big picture and now includes the *Team Demo* as part of that event. This will help reduce confusion between the Team Demo and System Demo events.
- Dev Team (/dev-team/), which consists of three to nine people, was added to the big picture. An Agile/Scrum Team in SAFe now consists of three roles: The Dev Team, Product Owner, and Scrum Master. Many Kanban teams may not use the Scrum Master or Product Owner roles, however, most find them useful.
- The Scrum (/scrumxp/)events (see Figure 22) were moved next to the ScrumXP icon . The font size is also easier to read and click.

Figure 22. Scrum Events

Simpler Glossary and Translations

The glossary (/glossary/) has been translated into *nine languages* and has a new dropdown widget (see Figure 23) to support

downloading both US letter and A4 .pdf formats. The glossary entry is now the first element of each SAFe article.

Figure 23. Glossary with dropdown for language selection

Combined SAFe Foundation Elements

As illustrated in Figure 24, SAFe's foundation contains the supporting principles, values, mindset, implementation guidance, and leadership roles needed to successfully deliver value at scale.

The foundation appeared in two places on the big picture—the left side and bottom. It has now been consolidated to one level at the bottom of the framework. Additional changes include:

- **Communities of Practice (CoP)** was relocated from the left-side foundation to the *spanning palette*.
- **Lean Agile Leaders** was moved from the left-side foundation to the bottom.
- **SAFe Program Consultant (SPC)** was added to the foundation. SPCs are a key part of the leadership that is needed to implement SAFe.
- **Implementation Roadmap** replaced Implementing 1-2-3.

Figure 24. SAFe 4.5 Foundation

Thanks to You, Our SAFe Community!

We owe a special thanks to you, our SAFe community, for all your feedback and support. Your passion for the best SAFe possible has driven us to deliver what we like to think of as the most adaptable—and now configurable—framework in the world. We do all this simply because 'better systems make the world a better place.'

Learn More

Read (http://dev.scaledagileframework.com/blog/safe-4-5-goes-live-top-5-reasons-to-update/) the blog post: 'SAFe 4.5 Goes Live: Top 5 Reasons to Update'

There are two new articles on the www.scaledagileframework.com site that provide an overview of SAFe:

- What is SAFe (/what-is-safe/) – provides a high-level overview of the framework.
- Why SAFe (/why-safe/) – describes why businesses need SAFe

[1] https://puppet.com/resources/whitepaper/2015-state-devops-report

[2] Ries, Eric. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Random House, Inc.

Last Update: 22 June 2017

Downloads (/posters/)

Latest Updates (/blog/)

TRAINING

Course Calendar (http://www.scaledagileacademy.com/events/event_list.asp)

About Certification (http://www.scaledagileacademy.com/?page=WhichCertification)

Become a Trainer (http://www.scaledagileacademy.com/?page=becomeatrainer)

PERMISSIONS

Permission FAQ (http://www.scaledagile.com/permission-faq/)

Permissions Form (http://www.scaledagile.com/permissions-form/)

Usage and Permissions (/usage-and-permissions/)

PARTNER

Becoming a Partner (http://www.scaledagile.com/become-a-partner/)

Partner Directory (http://www.scaledagile.com/listingcategory/directory/)

Partner Event Calendar (http://www.scaledagile.com/event-list/)

GET SOCIAL

Twitter (https://twitter.com/ScaledAgile)

Linkedin (https://www.linkedin.com/grps/Scaled-Agile-Framework-4189072?)

YouTube (https://www.youtube.com/user/scaledagile)

SlideShare (http://www.slideshare.net/ScaledAgile)

RECENT POSTS

Updated SAFe Scrum Master and SAFe Advanced Scrum Master courses now available! (/blog/updated-safe-scrum-master-and-safe-advanced-scrum-master-courses-now-available/)

Sep, 19th 2017

SAFe from the Trenches at Agile Australia (/blog/safe-from-the-trenches-at-agile-australia/)

Sep, 19th 2017

Updated SAFe Release Train Engineer and SAFe Product Owner/Product Manager courses now available! (/blog/updated-safe-release-train-engineer-and-safe-product-ownerproduct-manager-courses-now-available/)

Aug, 31th 2017

SCALED AGILE, INC

CONTACT US

5480 Valmont Rd., Suite 100
Boulder, CO 80301 USA

Email: support@scaledagile.com (mailto:support@scaledagile.com)

Phone: +1.303.554-4367

BUSINESS HOURS

Weekdays: 9am to 5pm
Weekends: CLOSED

(/)

> *"Many leaders pride themselves on setting the high-level direction and staying out of the details. But big picture, hands off leadership isn't likely to work in a change situation, because the hardest part of change—the paralyzing part—is in the details.*
>
> Any successful change requires a translation of ambiguous goals into concrete behaviors. To make a switch, you need to script the critical moves."
>
> —Dan and Chip, Heath, Switch: How to Change When Change Is Hard

SAFe Implementation Roadmap

This is the **home page** for the **SAFe® Implementation Roadmap series,** which consists of **twelve** articles.

The *SAFe Implementation Roadmap* consists of an overview graphic and a twelve-article series that describes a strategy and an ordered set of activities that have proven to be effective in successfully implementing SAFe.

Achieving the business benefits of Lean-Agile development at scale is not a trivial effort, so SAFe is not a trivial framework. Before realizing SAFe's rewards, organizations must embrace a Lean-Agile mindset, and understand and apply Lean-Agile principles. They must identify value streams and ARTs, implement a Lean-Agile portfolio, build quality in, and establish the mechanisms for continuous value delivery and DevOps. And, of course, the culture must evolve as well.

Based on proven organizational change management strategies, the SAFe Implementation Roadmap graphic and article series describe the steps, or "critical moves," an enterprise can take to implement SAFe in an orderly, reliable, and successful fashion.

Details

In order to achieve the desired organizational change, leadership must "script the critical moves" as described by Dan and Chip Heath[1]. When it comes to identifying those critical moves for adopting SAFe, hundreds of the world's largest enterprises have already gone down this path (see Case Studies (/case-studies/)) and successful adoption patterns have become clear. A fairly standard pattern is shown in Figure 1.

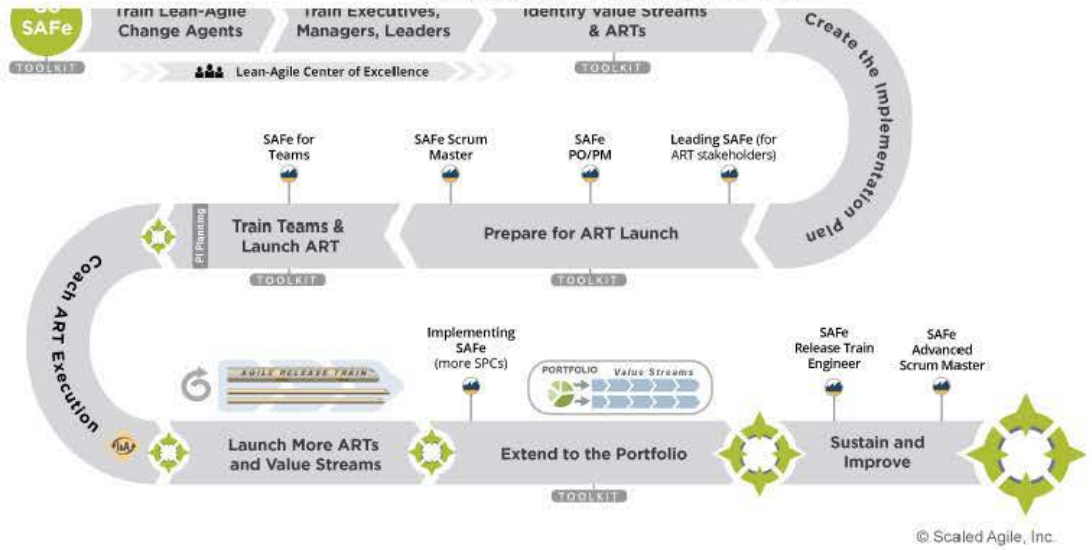


Figure 1. SAFe Implementation Roadmap

While no two adoptions are identical, and there is rarely a perfectly sequential step-by-step implementation in any enterprise, we know that businesses getting the best results typically follow a path similar to that shown in the Implementation Roadmap. It includes the following 12 steps:

1. Reaching the Tipping Point (/reaching-the-tipping-point/)
2. Train Lean-Agile Change Agents (/train-lean-agile-change-agents/)
3. Train Executives, Managers and Leaders (/train-executives-managers-and-leaders/)
4. Create a Lean-Agile Center of Excellence (/LACE/)
5. Identify Value Streams and ARTs (/identify-value-streams-and-arts/)
6. Create the Implementation Plan (/create-the-implementation-plan/)
7. Prepare for ART Launch (/prepare-for-art-launch/)
8. Train Teams and Launch ART (/train-teams-and-launch-the-art/)
9. Coach ART Execution (/coach-art-execution/)
10. Launch More ARTs and Value Streams (/launch-more-arts-and-value-streams/)
11. Extend to the Portfolio (/extend-to-the-portfolio/)
12. Sustain and Improve (/sustain-and-improve/)

This article serves as a launching pad for you to explore these steps in detail and understand how to apply them to your own implementation.

Moving Forward

Let's start with the first article:
Reaching the Tipping Point (/reaching-the-tipping-point/)

Best of luck to you and your SAFe Implementation. Stay true to these critical moves, and we are confident you will get the business benefits that you desire.

NEXT (/REACHING-THE-TIPPING-POINT/)

Learn More

[1] Heath, Chip; Heath, Dan. *Switch: How to Change Things When Change Is Hard*. The Crown Publishing Group

[2] Knaster, Richard and Leffingwell, Dean. SAFe Distilled, Applying the Scaled Agile Framework for Lean Software and Systems Engineering. Addison-Wesley, 2017.

Additional Resources

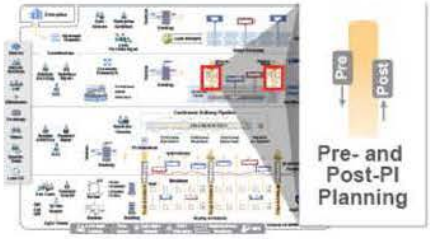
Download a SAFe Implementation Roadmap PDF

Download SAFe Implementation Roadmap PDF (http://www.scaledagileframework.com/?ddownload=35364)

Download a SAFe Implementation Roadmap image for PPT

Download SAFe Implementation Roadmap PPT (http://www.scaledagileframework.com/?ddownload=35363)

Last update: 22 May, 2017



(/)

> *Apply cadence, synchronize with cross-domain planning.*
>
> —SAFe Lean-Agile Principle #7

Pre- and Post-PI Planning

Pre- and Post-PI (Program Increment) planning events are used to prepare for, and follow-up after, PI Planning (/pi-planning/) for Agile Release Trains (/agile-release-train/) (ARTs) and Suppliers (/supplier/) in a Solution Train (/solution-train/).

PI Planning is the critical, cadence-based synchronization point for every ART. For large multi-ART Solutions, however, there are two additional activities: *Pre- and Post-PI Planning*. They support and coordinate the various ARTs involved in the Solution Train. Planning at this higher level helps align Solution (/solution/) building as a whole, and provides direction and visibility into where the train is going in the next Program Increment (/program-increment/) (PI).

While the timing and agenda for these meetings may vary based on Solution Context (/solution-context/), they typically occur just prior to, and just after, the ART planning sessions. The pre-PI planning meeting is used to set context for the upcoming ART PI planning sessions. Immediately afterwards, solution stakeholders participate in the ART planning sessions as well. The final meeting of the set is the post-PI planning session. Here, the results of ART planning are integrated into solution objectives for the upcoming PI, as well as the solution Roadmap (/roadmap/).

Details

Pre- and Post-PI Planning meetings allow Agile Release Trains (/agile-release-train/) and Suppliers (/supplier/) in large Value Streams (/value-streams/) to build a unified plan for the next Program Increment (PI) (/program-increment/). The pre- and post-PI planning meetings serve as the umbrella for the PI Planning (/pi-planning/) meetings at the Program Level (/program-level/), where the actual detailed planning takes place and is incorporated into the Innovation and Planning (IP) Iteration (/innovation-and-planning-iteration/) calendar. The pre-PI planning event is used to coordinate input objectives, key Milestones (/milestones/), and Solution Context (/solution-context/) and business context for the Agile Release Train planning sessions. The post-PI planning event is used to integrate the results of planning into the Vision (/vision/) and Roadmap (/roadmap/) for the value stream. At the end of the post-PI planning meeting, there should be an agreement on a set of value stream PI Objectives (/pi-objectives/) to be implemented by the end of the PI and demoed at the next Solution Demo (/solution-demo/). As with PI planning, the pre- and post-PI planning meetings deliver many business benefits:

- Provide open and productive communication through face-to-face alignment
- Synchronize ARTs with Solution Train vision via the ART and solution PI objectives
- Identify dependencies and foster cross-ART coordination
- Provide the opportunity for "just the right amount" of Solution (/solution/)-level architectural—and where applicable—User Experience guidance (see *Lean UX* (/lean-ux/))
- Match solution demand to ART capacities

Another benefit is team building across the solution train, which helps create the social fabric necessary to achieve high performance. In addition, as planning is based on known velocities, the post-PI planning meeting is a critical step in continuously assessing Work In Process (WIP) and in shedding excess WIP whenever necessary.

Inputs and Outputs

Inputs to pre- and post-PI planning include the solution roadmap, vision, Solution Intent (/solution-intent/), and the top Capabilities (/features-and-capabilities/) from the Solution Backlog (/program-and-solution-backlogs/). Attendees include:

- Customers (/customer/)
- Solution stakeholders—Solution Train Engineer (/release-train-engineer-and-solution-train-engineer/) (STE), Solution Management (/product-and-solution-management/), Solution Architect-Engineering (/system-and-solution-architect-engineering/), solution System Team (/system-team/), and Release Management (/release-management/)
- Representatives from all the ARTs and Suppliers, usually Product Management (/product-and-solution-management/), System Architect/Engineering (/system-and-solution-architect-engineering/), Release Train Engineers (/release-train-engineer-and-solution-train-engineer/) (RTEs), and engineering managers.

Outputs include three primary artifacts:

- A set of aggregated "SMART (https://en.wikipedia.org/wiki/SMART_criteria)" PI objectives for the solution
- A solution planning board, which highlights the objectives, anticipated delivery dates, and any other relevant milestones for the solution
- A vote of confidence/commitment to these objectives

This repetitive, "rolling-wave planning" process guides the solution through the inevitable technical obstacles and twists and turns of the business and technology environment.

Gain Context in the Solution Demo

The solution demo is to the solution train what the System Demo (/system-demo/) is to the ART. In this case, it's a regular opportunity to evaluate the fully integrated solution. Usually hosted by Solution Engineering, solution stakeholders typically attend. (This includes Solution Management and the Solution Train Engineer.) The insights from that meeting will inform these stakeholders of the current objective assessment of solution progress, performance, and potential fitness for use. While the timing of the solution demo will vary based on solution context, it provides critical objective inputs for the pre- and post-PI planning meetings.

Prepare for Pre- and Post-PI Planning

The pre- and post-PI planning meetings bring together stakeholders from all parts of the solution train. They require advance content preparation, coordination, and communication. The actual agendas and timelines listed below are a suggested way to run these meetings, but various value streams may adapt these to their own capabilities and locations.

Regardless of how the timing and physical logistics are actually arranged, all the elements of these meetings must happen to achieve true alignment across the trains and suppliers. It's critical that there is clear vision and context, that the right stakeholders participate, and that essential activities take place, including:

- **The executive briefing** – Defines current business, solution, and customer context
- **Solution vision briefing(s)** – Briefings prepared by Solution Management, including the *top capabilities* in the solution backlog
- **Milestones definitions** – Clearly explains upcoming key events and metrics

Set Planning Context in Pre-PI Planning

The pre-PI planning meeting is used to develop the context in which the ARTs and suppliers can create their plans. Individual sessions are described below, and a suggested overall agenda is shown in Figure 1.

Figure 1. Example pre-PI planning meeting agenda

- **PI Summary Reports** – Each ART and supplier presents a brief report of the accomplishments of the previous PI. This doesn't replace the solution demo, but it does provide the context of what has been achieved for the planning process.
- **Business Context and Solution Vision** – A senior executive presents a briefing about the current state of the solution and portfolio. Solution Management presents the current solution vision and highlights changes from the previous PI. They may also present the Roadmap for the upcoming three PIs, as well as milestones that fall during that period to ensure that they are known and addressed.
- **Solution Backlog** – Solution Management will review the top capabilities for the upcoming PI. Solution Architect-Engineering will discuss upcoming Enabler (/enablers/) Capabilities (/features-and-capabilities/) and Epics (/epic/).
- **Next PI Features** – Each ART's Product Management will present the Program Backlog (/program-and-solution-backlogs/) that they prepared for the upcoming PI and discuss dependencies with other trains.

Solution Stakeholders Participate in ART PI Planning

The practical logistics of large solution planning may limit all solution stakeholders from participating. However, it's important that key ART stakeholders—particularly Solution Management, the Solution Train Engineer (STE), and Solution Architect/Engineering—participate in as many of the ART PI planning sessions as possible. In many cases, ART planning sessions are largely concurrent, and these solution stakeholders participate by circulating among the ART PI planning sessions. Suppliers and customers play a critical role as well, and they should be represented in ART PI planning.

Summarize Results in Post-PI Planning

The post-PI planning meeting occurs after the ARTs have run their respective planning sessions, and is used to synchronize the ARTs and create the overall solution plan and roadmap. Participants include solution and key ART stakeholders. A sample agenda is shown in Figure 2. A discussion follows.

Figure 2. Example post-PI planning meeting agenda

- **PI Planning Report** – Each ART's Product Management presents the plans devised at their PI planning meetings, explaining the PI objectives and when each is anticipated to be available. Release Train Engineers (RTEs) fill out their ART's row of the solution planning board and discuss dependencies with other ARTs or suppliers.
- **Plan Review, Risk Analysis, and Confidence Vote** – All the participants review the complete plan. During PI planning, ARTs have identified critical risks and impediments that could affect their ability to meet their objectives. Relevant risks are addressed in a broader solution context in front of the full group. One by one, risks are categorized into the following groups, and addressed in a clear, honest, and visible manner:
  - **Resolved** – The group agrees that the issue is no longer a concern.
  - **Owned** – The item cannot be resolved in the meeting, so someone takes ownership

- **Owned** – The item cannot be resolved in the meeting, so someone takes ownership.
- **Accepted** – Some risks are facts or potential occurrences that simply must be understood and accepted.
- **Mitigated** – The group can identify a plan to mitigate the impact of an item.

Once all risks have been addressed, the group rates its confidence in meeting the solution's PI objectives. The team conducts a "fist-of-five vote." If the average is three or four fingers, then management should accept the commitment. If the average is fewer than three fingers, then adjustments are made and plans are reworked. Any person voting two fingers or fewer should be given time to voice their concern, which might add to the list of risks.

- **Plan Rework If Necessary** – If necessary, the group reworks its plans for as long as it takes for a commitment to be reached. This could cascade into follow-up meetings in the ARTs, as teams will need to be involved in any change to the plans.
- **Planning Retrospective and Moving Forward** – Finally, the STE leads a brief meeting retrospective to capture what went well, what didn't, and what could be done better next time. Then, next steps are discussed, including: capturing objectives, use of project management tooling, and finalizing the schedule of upcoming key activities and events.

Create the Right Outcomes

A successful event delivers three primary artifacts:

1. A set of "SMART" objectives for the solution train, with business value set by Solution Management, Solution Architect-Engineering, and customers. This may include stretch objectives, which are goals built into the plan but not committed to by the solution. Stretch objectives provide the flexible capacity and scope management options needed to increase reliability and quality of PI execution.
2. A solution planning board, which highlights the objectives, anticipated delivery dates, and any other relevant milestones, aggregated from the program boards, as Figure 3 illustrates.
3. A vote of confidence/commitment from the entire group regarding these objectives.

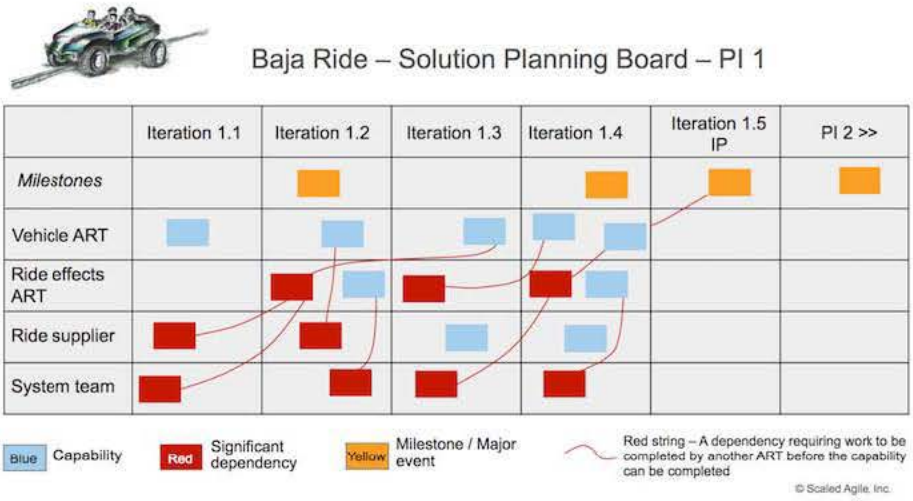


Figure 3. Example solution planning board

The solution roadmap is then updated based on the objectives for the planned PI.

Coordinating Solution Train and ART planning events in the IP iteration

Completing all the necessary refinement, pre- and post-planning, PI Planning, and I&A meetings in a solution train for the solution and all the member ARTs can represent a logistical challenge. Sequencing the meetings so that each has the right inputs and outputs in the optimal order requires precise planning by the STE and all RTEs to ensure the right stakeholders are present in the planning events. Figure 4 below shows one sample schedule of a two-week IP iteration with all the solution and ART events sequenced for the ideal flow of inputs and outputs for each meeting.

Figure 4. Example integrated calendar for solution train and ART events in an IP iteration

Note the two possible choices for the ART-level I&A workshops above. For geographically-distributed trains, scheduling I&A the day before PI planning is the most common pattern we've observed, as many train members may be traveling to a central location to participate. For cost-effective travel and venue scheduling, this requires a contiguous time block to conduct I&A and PI planning. When logistics are not the overriding consideration, the alternative is to conduct ART-level I&A events before the solution train I&A, as seen in the light shaded box with the dotted outline in week 1. This allows the outputs of the ART I&As—including demos, performance measures, and problem solving workshops—to flow into the solution train's I&A process as inputs. The benefit is that solution stakeholders will have the most current picture of the state of the solution going into their planning events.

Last update: 1 August, 2017

FRAMEWORK

Downloads (/posters/)

Latest Updates (/blog/)

TRAINING

Course Calendar (http://www.scaledagileacademy.com/events/event_list.asp)

About Certification (http://www.scaledagileacademy.com/?page=WhichCertification)

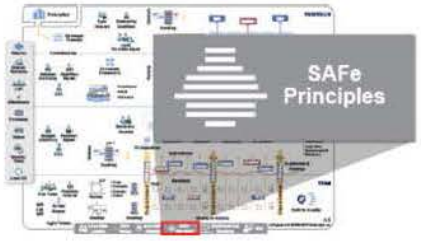
Become a Trainer (http://www.scaledagileacademy.com/?page=becomeatrainer)

PERMISSIONS

Permission FAQ (http://www.scaledagile.com/permission-faq/)

Permissions Form (http://www.scaledagile.com/permissions-form/)

Usage and Permissions (/usage-and-permissions/)



(/safe-lean-agile-principles)

> *Generate alternative system-level designs and subsystem concepts. Rather than try to pick an early winner, aggressively eliminate alternatives. The designs that survive are your most robust alternatives.*
>
> —Allen C. Ward, Lean Product and Process Development

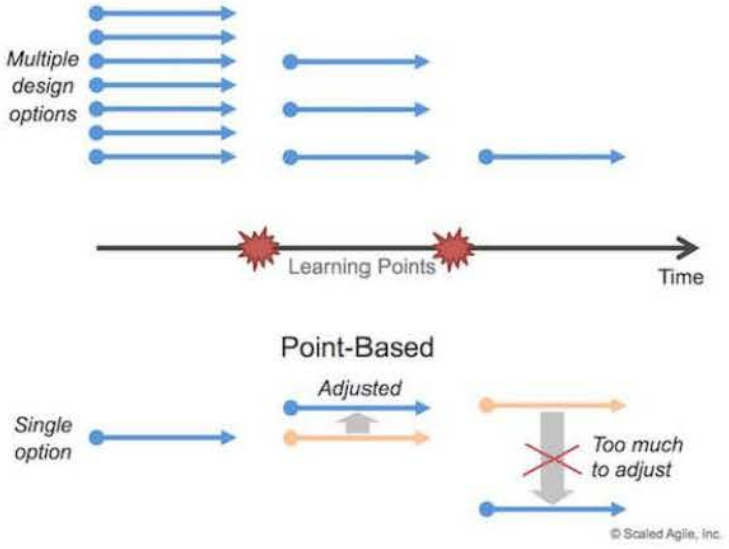
Principle #3 – Assume variability; preserve options

Systems builders tend to have a natural inclination to try to reduce variability. It just seems that the more you think you know and have already decided, the further along you are. But this is often not the case.

While it is true that variability can lead to bad outcomes, the opposite case can also be true.
Variability is not inherently bad or good. Rather, it is the economics associated with the timing and type of variability that determines the outcomes. A focus on eliminating variability too soon perpetuates a risk avoidance culture wherein people can't make mistakes and gain experience by learning what works and what doesn't.

Other than a general understanding of system intent, Lean systems builders recognize that very little is actually known at the beginning of the project. If it was, they would have already built it. However, traditional design practices tend to drive developers to quickly converge on a single option—a point in the potential solution space—and then modify that design until it eventually meets the system intent. This can be an effective approach, unless of course one picks the wrong starting point; then subsequent iterations to refine that solution can be very time consuming and lead to a suboptimal design [1]. And the bigger and more technically innovative the system is, the higher the odds are that your starting point was not the optimal one.

A better approach, referred to as Set-based Design or Set-based Concurrent Engineering [2], is illustrated in the figure below.



In this approach, the systems builder initially casts a wide net by considering multiple design choices at the start.

Thereafter, they continuously evaluate economic and technical tradeoffs—typically as exhibited by objective evidence presented at integration learning points. They then eliminate the weaker options over time; and finally, converge on a final design, based on the knowledge that has been gained to that point.

This process leaves design options open as long as possible, converges as and when necessary, and produces more optimal technical and economic outcomes.

Learn More

[1] Iansiti, Marco. "Shooting the Rapids: Managing Product Development in Turbulent Environments." *California Management Review* 38 (1995): 37–58.

[2] Ward, Allan C. and Durward Sobek. *Lean Product and Process Development*. Lean Enterprise Institute Inc., 2014.

Last update: 11 May 2015

FRAMEWORK

Downloads (/posters/)

Latest Updates (/blog/)

TRAINING

Course Calendar (http://www.scaledagileacademy.com/events/event_list.asp)

About Certification (http://www.scaledagileacademy.com/?page=WhichCertification)

Become a Trainer (http://www.scaledagileacademy.com/?page=becomeatrainer)

PERMISSIONS

Permission FAQ (http://www.scaledagile.com/permission-faq/)

Permissions Form (http://www.scaledagile.com/permissions-form/)

Usage and Permissions (/usage-and-permissions/)

PARTNER

Becoming a Partner (http://www.scaledagile.com/become-a-partner/)

Partner Directory (http://www.scaledagile.com/listingcategory/directory/)

Partner Event Calendar (http://www.scaledagile.com/event-list/)

GET SOCIAL

Twitter (https://twitter.com/ScaledAgile)

Linkedin (https://www.linkedin.com/grps/Scaled-Agile-Framework-4189072?)

YouTube (https://www.youtube.com/user/scaledagile)

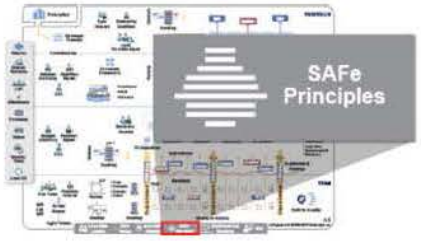
SlideShare (http://www.slideshare.net/ScaledAgile)

RECENT POSTS

Updated SAFe Scrum Master and SAFe Advanced Scrum Master courses now available! (/blog/updated-safe-scrum-master-and-safe-advanced-scrum-master-courses-now-available/)

Sep, 19th 2017

SAFe from the Trenches at Agile Australia (/blog/safe-from-the-trenches-at-agile-australia/)



(/safe-lean-agile-principles)

> *There was in fact no correlation between exiting phase gates on time and project success ... the data suggested the inverse might be true.*
>
> —Dantar P. Oosterwal, The Lean Machine

Principle #5 – Base milestones on objective evaluation of working systems

The Problem with Phase-Gate Milestones

The development of today's large systems requires substantial investment—an investment that can reach millions, tens of millions, and even hundreds of millions of dollars. Together, systems builders and customers have a fiduciary responsibility to ensure that the investment in new solutions will deliver the necessary economic benefit. Otherwise, there is no reason to make the investment.

Clearly, stakeholders must collaborate in such a way as to help ensure the prospective economic benefit *throughout* the development process and not just engage in "wishful thinking" that all will be well at the end. To address this challenge, the industry has generally applied a sequential phase-gated (waterfall) development process, whereby progress is measured—and control is exercised—via a series of specific milestones.

These phase-gate milestones are not arbitrary. They follow the apparently logical and sequential process of discovery, requirements, design, development, test, and delivery. Of course it hasn't worked out all that well for many, as Figure 1 shows.

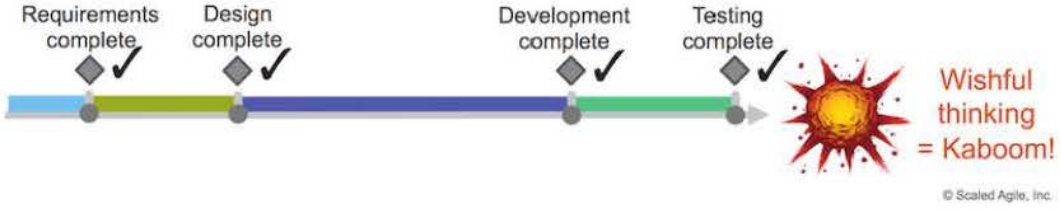


Figure 1. The problem with phase-gate milestones

The root causes of this problem is the failure to recognize *four critical errors* within the basic assumption that phase gates reveal real progress and thereby mitigate risk:

1. Centralizing requirements and design decisions in siloed functions that may not be integrally involved in the solution building.
2. Forcing too-early design decisions and "false positive feasibility" [1]: An early choice is made for the best known option at that time; development proceeds under the assumption that everything is on track; only later comes the discovery that the path chosen is not actually feasible. (Principle #3)
3. Assuming a "point" solution exists and can be built right the first time. This ignores the variability inherent in the process and provides no legitimate outlet for it. Variability will find a way to express itself.

4. Taking up-front decisions creates large batches of requirements, code and tests, and long queues. This leads to large batch handoffs and delayed feedback. (Principle #6)

Base Milestones on Objective Evidence

Clearly, the phase gate model does not mitigate risk as intended, and a different approach is needed. Principle #4 – ***Build incrementally with fast, integrated learning cycles*** provides elements of a solution to this dilemma.

Throughout development, the system is built in increments, each of which is an integration point that demonstrates some evidence as to the viability of the current in-process solution. Unlike phase-gate development, every milestone involves a portion of each step—requirements, design, development, testing—together producing an increment of value (see Figure 2). Further, this is done routinely, on a *cadence* (Principle #7), which provides the discipline needed to ensure periodic availability and evaluation, as well as predetermined time boundaries that can be used to collapse the field of less desirable options.

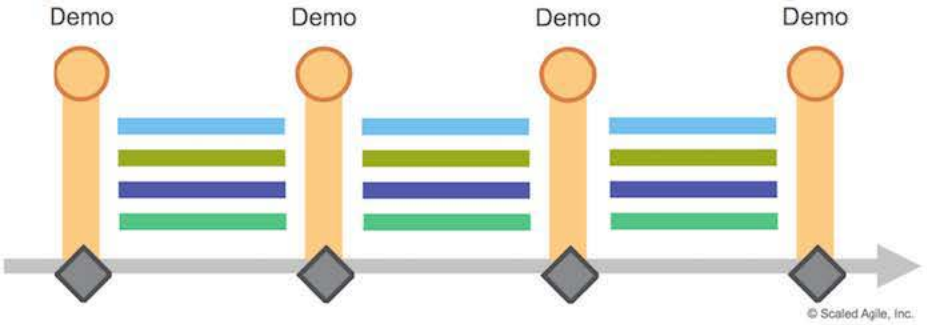


Figure 2. Milestones based on objective evaluation of working systems

What is actually measured at these critical integration points is subject to the nature and type of the system being built. But the system can be measured and assessed, and evaluated by the relevant stakeholders *frequently, and throughout the solution development life cycle*. This provides the financial, technical, and fitness-for-purpose governance needed to ensure that the continuing investment will produce a commensurate return.

Learn More

[1] Oosterwal, Dantar P. *The Lean Machine: How Harley-Davidson Drove Top-Line Growth and Profitability with Revolutionary Lean Product Development*. Amacom, 2010.

Last update: 11 April 2016

Permission FAQ (http://www.scaledagile.com/permission-faq/)

Permissions Form (http://www.scaledagile.com/permissions-form/)

Usage and Permissions (/usage-and-permissions/)

PARTNER

Becoming a Partner (http://www.scaledagile.com/become-a-partner/)

Partner Directory (http://www.scaledagile.com/listingcategory/directory/)

Partner Event Calendar (http://www.scaledagile.com/event-list/)

GET SOCIAL

Twitter (https://twitter.com/ScaledAgile)

Linkedin (https://www.linkedin.com/grps/Scaled-Agile-Framework-4189072?)

YouTube (https://www.youtube.com/user/scaledagile)

SlideShare (http://www.slideshare.net/ScaledAgile)

RECENT POSTS

Updated SAFe Scrum Master and SAFe Advanced Scrum Master courses now available! (/blog/updated-safe-scrum-master-and-safe-advanced-scrum-master-courses-now-available/)

Sep, 19th 2017

SAFe from the Trenches at Agile Australia (/blog/safe-from-the-trenches-at-agile-australia/)

Sep, 19th 2017

Updated SAFe Release Train Engineer and SAFe Product Owner/Product Manager courses now available! (/blog/updated-safe-release-train-engineer-and-safe-product-ownerproduct-manager-courses-now-available/)

Aug, 31th 2017

SCALED AGILE, INC

CONTACT US

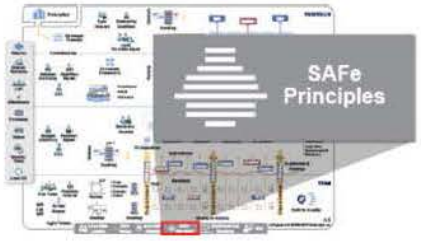
5480 Valmont Rd., Suite 100
Boulder, CO 80301 USA

Email: support@scaledagile.com (mailto:support@scaledagile.com)

Phone: +1.303.554-4367

BUSINESS HOURS

Weekdays: 9am to 5pm
Weekends: CLOSED



(/safe-lean-agile-principles)

> *Knowledge workers themselves are best placed to make decisions about how to perform their work.*
>
> —Peter F. Drucker

Principle #9 – Decentralize decision-making

Delivering value in the shortest sustainable lead time requires decentralized decision-making. Any decision that must be escalated to higher levels of authority introduces a delay in delivery. In addition, escalated decisions can decrease decision fidelity due to the lack of local context, plus changes to fact patterns that occur during the waiting period.

Simply, decentralized decision-making reduces delays, improves product development flow and throughput, and enables faster feedback and more innovative solutions. Higher levels of empowerment are an additional, material benefit.

Centralize Strategic Decisions

However, this is not to say, that all decisions should be decentralized. Some decisions are strategic, have far reaching impact and are largely outside of the purview of the teams. After all, leaders still have accountability for outcomes, and they have the market knowledge, longer-range perspectives, and understanding of the business and financial landscape necessary to steer the enterprise.
This leads us to an understanding that some decisions should be centralized. Generally these decisions have the following characteristics:

- *Infrequent*. These decisions aren't made very often, and typically are not urgent. Deeper consideration is appropriate. (examples: product strategy, international commitments).
- *Long Lasting*. Once made, these decisions are unlikely to change. (Examples: commitment to a standard technology platform. Commit to organizational realignment around value streams.)
- *Provide significant economies of scale*. These decisions provide large and broad economic benefit. (Examples: a common way of working, standard development languages, standard tooling, offshoring)

Leadership is charged with making these types of decisions, supported by the input of those impacted by the decision.

Decentralize Everything Else

The vast majority of decisions do not reach the threshold of strategic importance. All other decisions should be decentralized. Characteristics of these types of decisions include:

- *Frequent*. These decisions are frequent and common. (examples: team and program backlog prioritization, real-time ART scoping, response to defects and emerging issues)
- *Time critical*. A delay in these types of decisions comes at a high cost of delay. (Examples: point releases, customer emergencies, dependencies with other teams)
- *Require local information*. These decisions need specific local context, whether it be technology, organization, or specific customer or market impact (Examples: ship/no ship release to a specific customer, resolve a significant design problem, self-organization of individuals and teams to an emerging challenge)

These decisions should be made by the workers who have better local context and detailed knowledge of the technical complexities of the current situation.

A lightweight thinking tool for decision making

Understanding how decisions are made is a key factor in empowering knowledge workers. Leadership's responsibility is to establish the rules for decision making (including, for example, the economic framework), and then largely empower others to make them. A simple tool and exercise for thinking about whether decisions should be centralized or de-centralized is provided in Figure 1.

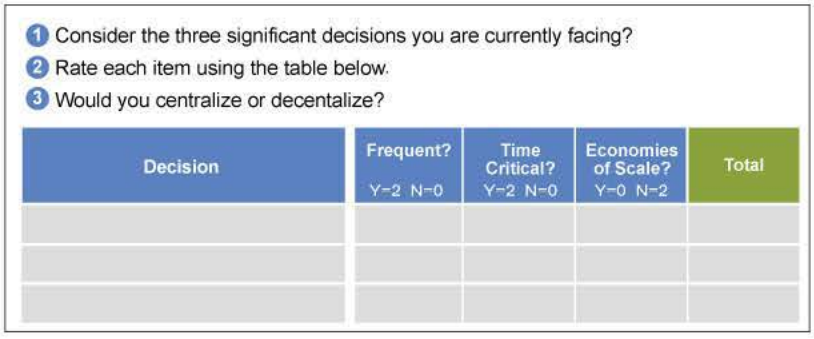


Figure 1. A simple decision-making framework and exercise

Last update: 15 August 2016

(/)

> *Decentralize decision-making*
>
> —SAFe Lean-Agile Principle #9

Product and Solution Management

The *Product Management* role has content authority for the Program Backlog (/program-and-solution-backlogs/). They are responsible for identifying customer needs, prioritizing features and developing the program Vision (/vision/) and Roadmap (/roadmap/).

In a similar manner, Solution Management has the content authority for the Solution Backlog (/program-and-solution-backlogs/). They work with Customers to understand their needs, create the Solution vision and Roadmap, define requirements, and guide work through the Solution Kanban (/program-and-solution-kanbans/).

Lean-Agile enterprises focus on delivering the right Solutions to customers with the highest quality in the shortest lead times. This requires people with clear *content authority* to take responsibility for continuously defining, prioritizing, and validating requirements. Working closely with development in short, integrated learning cycles, they bring the voice of the customer to the developers, and the voice of development to the customer.

In accordance with Principle 9 – *Decentralized Decision-Making*, SAFe prescribes a content authority chain that extends through three levels. Solution Management (/product-and-solution-management/) is responsible for guiding the solutions that constitute the Solution. Product Management is responsible for Program Vision and Backlog. And, finally, Product Owners (/product-owner/) make fast, local content decisions on behalf of the Agile Team.

This article describes the important roles that Product Managers and Solutions Managers play in SAFe. While the roles are similar in most respects, they operate at different levels of the solution and manage different concerns.

Details

Product Management and *Solution Management* are the main *content authority* for the Program and Solution Backlogs, respectively. They create the Vision (/vision/) and work with Customers (/customer/) to understand their needs, define requirements, and guide work through the Solution and Program Kanbans (/program-and-solution-kanbans/). They prioritize work using WSJF (/wsjf/), schedule Feature and Capabilities for Release (/release-on-demand/) using the Roadmap (/roadmap/), validate customer response, and provide fast feedback.

A Lean-Agile Approach to Content Management

What SAFe describes as *content* has traditionally been represented by a Marketing Requirements Document (MRD), Product Requirements Document, and System and Software Requirements Specifications (SRS).

In waterfall development, these specifications were typically done up front, with an expectation that all the requirements could be established in advance of Solution (/solution/) development. However, it didn't work out all that well that way, and the move to a Lean-Agile approach is driven in large part by that result.

We now understand that assumptions about requirements, design, and architecture all need to be validated through actual development, testing, and experimentation [1], and that teams must be open to emerging knowledge that can be quickly fed back into the solution. In SAFe, Continuous Exploration (/continuous-exploration/) is the process used to continually explore market and user needs, and defining a Vision, Roadmap, and set of features and capabilities that address those needs.

As we see in Solution Intent (/solution-intent/), some of the requirements of the solution are likely to be well understood and fixed from the beginning, while others are variable and can only be understood during the development process. Managing this new approach is the primary responsibility of Product and Solution Management. In the Lean Enterprise (/enterprise/), these responsibilities must be fulfilled in a far more Agile manner, as is illustrated in Table 1.

Responsibility	Traditional	Agile
Understand customer need	Up-front and discontinuous	Constant interaction with the Customer, who is part of the value stream. Other techniques include: customer visits, gemba walks, elicitation (e.g. such as interviews and surveys, brainstorming, trade studies, market research)
Document requirements	Fully elaborated in documents; handed off	High-level vision; constant product and solution backlog refinement and informal face-to-face communication with Agile Teams
Schedule	Created in hard-committed Roadmaps and Milestones at the beginning	Continuous near-term roadmapping
Prioritize requirements	Not at all, or perhaps one-time only, often in requirements document form	Reprioritized at every PI boundary via WSJF; constant scope triage
Validate requirements	Not applicable; QA responsibility	Primary role, involved with Iteration and PI System Demos; acceptance criteria included; fitness for purpose understood
Manage delivery schedule	Typically one time, fixed well in advance	Released frequently, whenever there is enough value
Manage change	Change avoided—weekly change control meetings	Change embraced; adjusted at PI and Iteration boundaries

Table 1. Changes in Product and Solution Management behavior in a Lean-Agile enterprise

Responsibilities of Product Management

The following section describes the primary responsibilities of the Product Manager in the context of a single Agile Release Train (/agile-release-train/). For larger (multiple-ART) Solution Trains, additional responsibilities are necessary. Those are described in later sections.

- **Understand Customer needs and validate solutions.** Product Management is the internal voice of the Customer for the ART and works with Customers (as well as Product Owners (/product-owner/)) to constantly understand and communicate their needs and participate in validation of the proposed solutions.
- **Understand and support portfolio work.** Every ART lives in the context of a portfolio, so Product Management has a responsibility to understand the Budget (/lean-budgets/) parameters for the upcoming fiscal period, understand how Strategic Themes (/strategic-themes/) influence the strategic direction, and work with Epic Owners (/epic-owner/) to develop the business case for Epics (/epic/) that affect their ART.
- **Develop and communicate the program vision and roadmap.** Product Management continuously develops and communicates the vision to the development teams and defines the Features (/features-and-capabilities/) of the system. In collaboration with (/system-and-solution-architect-engineering/)System and Solution Architect/Engineering (/system-and-solution-architect-engineering/), they also define and maintain the Nonfunctional Requirements (NFRs) (/nonfunctional-requirements/), to help ensure that the solution meets relevant standards and other system quality requirements. They are responsible for the roadmap, which illustrates, at a high level, how features are intended to be implemented over time.
- **Manage and prioritize the flow of work.** Product Management manages the flow of work though the program Kanban and into the program backlog. Product Management is responsible for making sure that there are enough *ready* features in the backlog at all times. To be ready, they develop feature *acceptance criteria* that can be used to establish that the feature meets its Definition of Done (DoD). And since judicious selection and sequencing of features is the key economic driver for each ART, the backlog is reprioritized with WSJF prior to each PI Planning (/pi-planning/) session.
- **Participate in PI planning.** During each PI planning session, Product Management presents the vision, which highlights the proposed features of the solution, along with any relevant upcoming Milestones (/milestones/). They also typically participate as Business Owners (/business-owners/)for the train, with the responsibility of approving PI Objectives (/PI-objectives/) and establishing business value.
- **Define releases and Program Increments.** Owning the '*what*' means that Product Management is largely responsible for release definition as well, including new features, architecture, and allocations for technical debt. This is accomplished

release definition as well, including new features, architecture, and allocations for technical debt. This is accomplished through a series of Program Increments (/program-increment/)and releases, whose definition and business objectives are also determined by Product Management. Product Management works with Release Management (/release-on-demand/), where applicable, to decide when enough value has been accrued to warrant a release to the Customer.

- **Work with System Architect/Engineering to understand Enabler work.** While Product Management is not expected to drive technological decisions, they are expected to understand the scope of the upcoming Enabler (/enablers/) work and to work with System and Solution Architect/Engineering to assist with decision-making and sequencing of the key technological infrastructures that will host the new business functionality. This can often best be accomplished by establishing a capacity allocation, as described in Program Backlog (/program-and-solution-backlogs/).
- **Participate in demos and Inspect and Adapt.** Product Management is an active participant in biweekly System Demos (/system-demo/), including the aggregate one at the end of the PI. They are also involved in assessment of Metrics (/metrics/), including evaluation of business value achieved versus plan, and are active participants in the Inspect and Adapt (/inspect-and-adapt/) workshop.
- **Build an effective Product Manager/Product Owner team.** Though the Product Owner and Product Management roles may report to different organizations, forming an effective extended Product Management/Product Owner team is the key to efficient and effective development. Such a team also contributes materially to the job satisfaction that comes with being part of a high-performing team, one that routinely delivers on its quality and vision commitments.

Product Management's Participation in Large Value Streams

The above section highlights the role of Product Management in the context of the ART. For teams building large solutions that require multiple ARTs, Product Management has additional responsibilities:

- **Collaborate with Solution Management.** At the large solution level, Solution Management plays a similar role, but with a focus on the capabilities of the larger solution. But building an effective solution is no more effective than the collaboration between the two roles. This collaboration involves participation in solution backlog refinement and prioritization, as well as splitting Capabilities (/features-and-capabilities/) into features, and NFRs, as the case may be.
- **Participate in Pre- and Post-PI Planning.** Product Management also participates in the Pre-PI Planning (/pre-and-post-pi-planning/) meeting, working with the value stream stakeholders to define the inputs, milestones, and high-level objectives for the upcoming PI planning session. In the Post-PI Planning (/pre-and-post-pi-planning/)session, Product Management helps synthesize findings into an agreed-to set of solution PI objectives.
- **Participate in the Solution Demo.** Product Management participates in the Solution Demo (/solution-demo/), often demonstrating the capabilities that their ART has contributed and reviewing the contributions of the other ARTs, always with a systems view, and always with an eye toward fitness of purpose.
- **Collaborate with Release Management.** In larger-scale systems, Release Management also plays a significant role. Product Management works with the key stakeholders on progress, budget, release strategy, and releasability of their elements of the solution.

Responsibilities of Solution Management

Solution Management plays much the same role that Product Management plays, but at the large solution level. There, Solution Management is part of a critical trio—Solution Management, Solution Train Engineer (/release-train-engineer-and-solution-train-engineer/), and Solution Architect/Engineering (/system-and-solution-architect-engineering/)—that shares much of the responsibility for solution success. Solution Management is responsible for the solution intent, which captures and documents fixed and variable solution level behaviors. They also work with Release Management where applicable.

Responsibilities include working with portfolio stakeholders, Customers, ARTs and Solution Trains to understand needs and build and prioritize the solution backlog. They have similar vision, roadmap, solution Kanban, and solution demo activities as well.

Solution Management plays a crucial role in pre- and post-PI planning, as well as Large Solution-level inspect and adapt workshops. They also work with Suppliers (/supplier/), making sure the requirements for supplier-delivered capabilities are understood and assisting with the conceptual integration of these concerns.

Learn More

[1] Ries, Eric. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business, 2011.

[2] Leffingwell, Dean. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison-Wesley, 2011.

Last update: 17 June, 2017

FRAMEWORK

Downloads (/posters/)

Latest Updates (/blog/)

TRAINING

Course Calendar (http://www.scaledagileacademy.com/events/event_list.asp)

About Certification (http://www.scaledagileacademy.com/?page=WhichCertification)

Become a Trainer (http://www.scaledagileacademy.com/?page=becomeatrainer)

PERMISSIONS

Permission FAQ (http://www.scaledagile.com/permission-faq/)

Permissions Form (http://www.scaledagile.com/permissions-form/)

Usage and Permissions (/usage-and-permissions/)

PARTNER

Becoming a Partner (http://www.scaledagile.com/become-a-partner/)

Partner Directory (http://www.scaledagile.com/listingcategory/directory/)

Partner Event Calendar (http://www.scaledagile.com/event-list/)

GET SOCIAL

Twitter (https://twitter.com/ScaledAgile)

Linkedin (https://www.linkedin.com/grps/Scaled-Agile-Framework-4189072?)

YouTube (https://www.youtube.com/user/scaledagile)

SlideShare (http://www.slideshare.net/ScaledAgile)

RECENT POSTS

Updated SAFe Scrum Master and SAFe Advanced Scrum Master courses now available! (/blog/updated-safe-scrum-master-and-safe-advanced-scrum-master-courses-now-available/)

Sep, 19th 2017

SAFe from the Trenches at Agile Australia (/blog/safe-from-the-trenches-at-agile-australia/)

Sep, 19th 2017

Updated SAFe Release Train Engineer and SAFe Product Owner/Product Manager courses now available! (/blog/updated-safe-release-train-engineer-and-safe-product-ownerproduct-manager-courses-now-available/)

Aug, 31th 2017

SCALED AGILE, INC

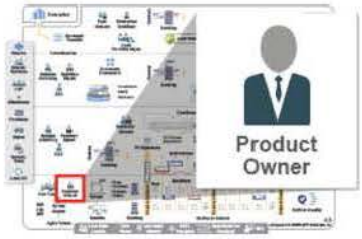
CONTACT US

5480 Valmont Rd., Suite 100
Boulder, CO 80301 USA

Email: support@scaledagile.com (mailto:support@scaledagile.com)

Phone: +1.303.554-4367

BUSINESS HOURS



(/)

> *Business people and developers must work together daily throughout the project.*
>
> —Agile Manifesto

Product Owner

The *Product Owner (PO)* is the content authority for the team level. They are responsible for the team backlog, prioritizing and accepting stories, and representing the customer to the Agile team.

The PO is the team member responsible for defining Stories and prioritizing the Team Backlog, streamlining the execution of program priorities, while maintaining conceptual and technical integrity of the Features or components the team is responsible for. The PO has a significant role in quality and is the only team member empowered to accept stories as done. For most enterprises converting to Agile, this is a new and *critical* role, which typically translates into a full-time job. It requires one PO to support each Agile Team (or, at most, two teams).

The role has significant relationships and responsibilities outside the local team, including working with Product Management to prepare for Program Backlog, participating in ART Synch meetings, and cooperating to deliver Solution value.

Details

The *Product Owner* serves as the Customer (/customer/) proxy and is responsible for working with Product Management (/product-and-solution-management/) and other stakeholders—including other Product Owners—to define and prioritize Stories (/story/) in the Team Backlog (/team-backlog/) so that the Solution (/solution/) effectively addresses program priorities (Features (/features-and-capabilities/)/Enablers (/enablers/)) while maintaining technical integrity. Ideally, the Product Owner is collocated with the rest of the team, where they typically share management, incentives, and culture. But the Product Owner also attends most relevant Product Management meetings about planning and Program Backlog (/program-and-solution-backlogs/)/Vision (/vision/) refinement.

Responsibilities

The SAFe Product Owner fulfills the primary responsibilities outlined below.

Preparation and Participation in PI Planning

- As a member of the extended Product Management team, the Product Owner is heavily involved in Program Backlog (/program-and-solution-backlogs/) refinement and preparation for PI Planning (/PI-planning/) and also has a significant role in the planning event itself. Prior to the event, the Product Owner updates the team backlog and typically participates in reviewing and contributing to the vision, Roadmap (/roadmap/), and content presentations.
- During the event, the Product Owner is involved with story definition, providing clarifications necessary to assist the team with their story estimates and story sequencing, and drafting the team's specific objectives for the upcoming Program Increment (PI) (/program-increment/).

Iteration Execution

- *Backlog Refinement*. With input from System Architect/Engineering (/system-and-solution-architect-engineering/) and other stakeholders, the Product Owner has the primary responsibility for building, pruning, and maintaining the team backlog. The backlog consists mostly of user stories but also includes defects and enablers. Backlog items are prioritized based on user value, time, and other team dependencies that are determined in the PI planning meeting and refined during the PI.
- *Iteration Planning*. The Product Owner reviews and reprioritizes the backlog as part of the preparatory work for iteration (/iterations/) planning (see "Plan the Iteration" in ScrumXP (/iteration-planning/)), including coordination of content dependencies with other Product Owners. During the iteration planning meeting, the product owner is the main source for story detail and priorities and has the responsibility of accepting the final iteration plan.
- *Just-In-Time Story Elaboration*. Most backlog items are elaborated into user stories for implementation. This may happen *prior* to the iteration, *during iteration planning*, or *during the iteration* itself. While any team member can write stories and acceptance criteria, the Product Owner has the primary responsibility for keeping the process flowing. It is usually good to have approximately two iterations' worth of ready stories in the team backlog at all times. More would create a queue, while less might inhibit flow.
- *Supporting ATDD*. POs participate in development of story acceptance criteria, draft them when feasible, and provide examples in support of ATDD (acceptance test-driven development) specification by example. See Test-First (/test-first/).
- *Accepting Stories*. The PO is the only team member who can accept stories as done. This includes validation that the story meets acceptance criteria and has the appropriate, persistent acceptance tests, and that it otherwise meets its Definition of Done. In so doing, the PO also fulfills a quality assurance function, focusing primarily on fitness for use.
- *Understand Enabler Work*. While Product Owners are not expected to drive technological decisions, they are expected to understand the scope of the upcoming enabler work and to work with System and Solution Architect/Engineering (/system-and-solution-architect-engineering/) to assist with decision-making and sequencing of the key technological infrastructures that will host the new business functionality. This can often best be accomplished by establishing a capacity allocation, as described in the *Team Backlog (/team-backlog/)* article.
- *Participate in Iteration Review and Retrospective*. As an integral member of the team and the one responsible for requirements, the PO has an important role in the Iteration Review (/iteration-review/), reviewing and accepting stories; and in the Iteration retrospective (see "Retrospect and Improve" in ScrumXP (/iteration-retrospective/)), where the teams gather to improve their processes. POs are also active participants in the ART's Inspect and Adapt (/inspect-and-adapt/) workshop.

Program Execution

- Iterations and teams both serve a larger purpose: frequent, reliable, and continuous release of value-added solutions. During the course of each PI, the Product Owner coordinates content dependencies with other Product Owners. This is often accomplished in part by attendance at weekly PO sync meetings. See the *Program Increment (/program-increment/)*article for more information.
- The Product Owner also has an instrumental role in producing the System Demo (/system-demo/) for program and Value Stream (/value-streams/) stakeholders.

Inspect and Adapt

- Teams address their larger impediments in the PI Inspect and Adapt workshop. There, the Product Owner works across teams to define and implement improvement stories that will increase the velocity and quality of the program.
- The PI system demo occurs as part of the Inspect and Adapt workshop. The Product Owner has an instrumental role in producing the PI system demo for program stakeholders.
- POs also participate in the preparation of the PI system demo to make sure that they will be able to show the most critical aspects of the solution to the stakeholders.

Content Authority

At scale, a single person cannot handle product and market strategy while also being dedicated to an Agile Team. Since Product Management and the Product Owner share the "content authority" for the program, it is important to have a clear delineation of roles and responsibilities, as is illustrated in Figure 1 below.

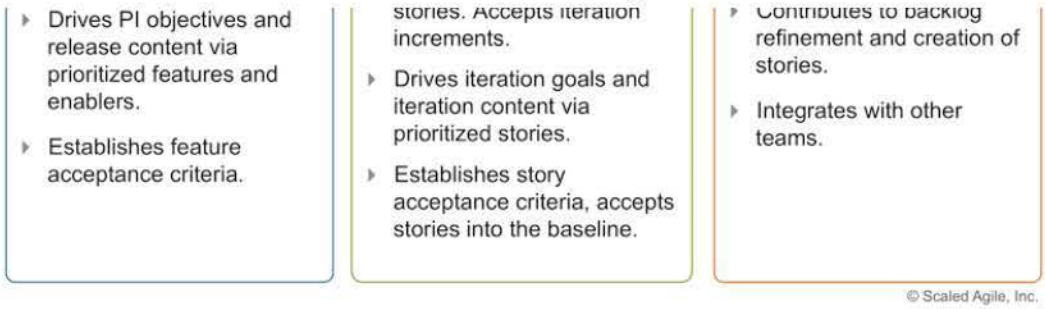


Figure 1. Release content governance

Fan-out Model of Product Manager, Product Owner, and Agile Teams

Successful development is, in part, a game of numbers in the Enterprise (/enterprise/). Without the right number of people in the right roles, bottlenecks will severely limit velocity. Therefore, the number of Product Managers, Product Owners, and Agile Teams must be roughly in balance in order to properly steer the Agile Release Train (ART) (/agile-release-train/), or the whole system will spend much of its time *waiting* for definition, clarification, and acceptance. The Framework recommends a fan-out, as illustrated in Figure 2.

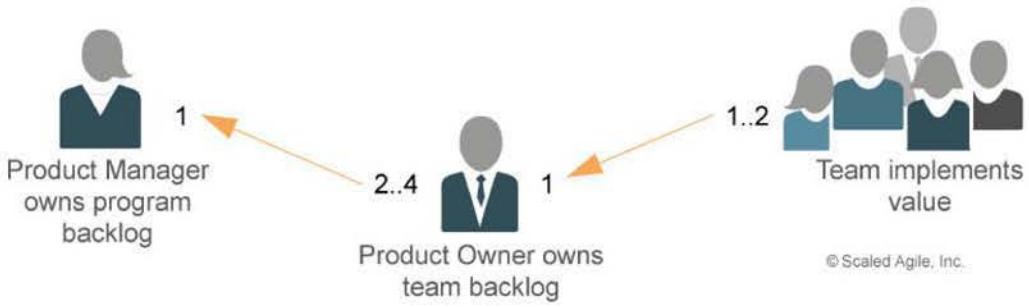


Figure 2. Fan-out model for PO/PM

Each Product Manager can usually support up to four Product Owners, each of whom can be responsible for the backlog for one or two Agile Teams.

Learn More

[1] Leffingwell, Dean. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise.* Addison-Wesley, 2011.

[2] Larman, Craig, and Bas Vodde. *Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum.* Addison-Wesley, 2010.

Last update: 21 June, 2017

About Certification (http://www.scaledagileacademy.com/?page=WhichCertification)

Become a Trainer (http://www.scaledagileacademy.com/?page=becomeatrainer)

PERMISSIONS

Permission FAQ (http://www.scaledagile.com/permission-faq/)

Permissions Form (http://www.scaledagile.com/permissions-form/)

Usage and Permissions (/usage-and-permissions/)

PARTNER

Becoming a Partner (http://www.scaledagile.com/become-a-partner/)

Partner Directory (http://www.scaledagile.com/listingcategory/directory/)

Partner Event Calendar (http://www.scaledagile.com/event-list/)

GET SOCIAL

Twitter (https://twitter.com/ScaledAgile)

Linkedin (https://www.linkedin.com/grps/Scaled-Agile-Framework-4189072?)

YouTube (https://www.youtube.com/user/scaledagile)

SlideShare (http://www.slideshare.net/ScaledAgile)

RECENT POSTS

Updated SAFe Scrum Master and SAFe Advanced Scrum Master courses now available! (/blog/updated-safe-scrum-master-and-safe-advanced-scrum-master-courses-now-available/)

Sep, 19th 2017

SAFe from the Trenches at Agile Australia (/blog/safe-from-the-trenches-at-agile-australia/)

Sep, 19th 2017

Updated SAFe Release Train Engineer and SAFe Product Owner/Product Manager courses now available! (/blog/updated-safe-release-train-engineer-and-safe-product-ownerproduct-manager-courses-now-available/)

Aug, 31th 2017

SCALED AGILE, INC

CONTACT US

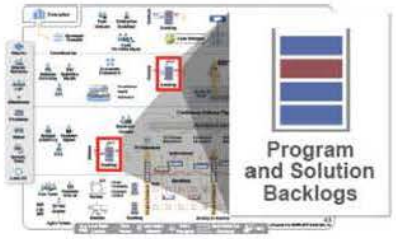
5480 Valmont Rd., Suite 100
Boulder, CO 80301 USA

Email: support@scaledagile.com (mailto:support@scaledagile.com)

Phone: +1.303.554-4367

BUSINESS HOURS

Weekdays: 9am to 5pm
Weekends: CLOSED



(/)

> *The emphasis should be on why we do a job.*
>
> —W. Edwards Deming

Program and Solution Backlogs

The *Program Backlog* is the final state in the Program Kanban, which is also the last state of Continuous Exploration. It's the holding area for a prioritized list of Features that have been analyzed and are intended to address user needs and deliver business benefits for a single Agile Release Train (ART). It also contains the Enabler features necessary to build the Architectural Runway.

The *Solution Backlog* is the holding area for upcoming Capabilities and Solution enablers, each of which can span multiple ARTs, and are intended to advance the Solution and build its architectural runway.

Managing the backlog is the responsibility of Product Management (/product-and-solution-management/) (Program Backlog) and Solution Management (/product-and-solution-management/) (Solution Backlog). The items in the backlog result from research activities and an active collaboration with various stakeholders—Customers (/customer/), Business Owners (/business-owners/), Product Management (/product-and-solution-management/), Product Owners (/product-owner/), Architects (/system-and-solution-architect-engineering/), and more. They travel through the Kanban states of 'funnel' and 'analyzing.' Effectively identifying, refining, prioritizing, and sequencing backlog items using WSJF is the key to the economic success of the solution.

Since the backlog contains both new business functionality and the enablement work necessary to extend the architectural runway, capacity allocation (see Figure 3) is used to help ensure immediate and long-term value delivery, with velocity and quality.

Details

The *Program and Solution Backlogs* are the repositories for all the upcoming work that affects the behavior of the Solution (/solution/). Product and Solution Management (/product-and-solution-management/) develop, maintain, and prioritize the program and solution backlogs respectively. The backlogs are a short-term holding area for Features and Capabilities (/features-and-capabilities/) that have gone through the Program and Solution Kanbans (/program-and-solution-kanbans/) and have been approved for implementation. Backlog items are estimated in story points, as Figure 1 illustrates.

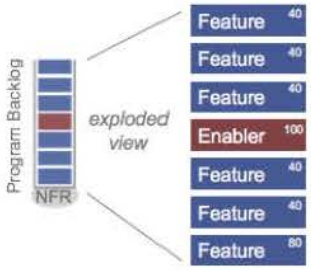


Figure 1. Exploded view of program backlog, with story point size estimates

Refining the Backlog

Agile Release Trains (ARTs) (/agile-release-train/) and Solution Trains (/solution-train/) run a steady eight to twelve 12 week Program Increment (/program-increment/) cadence of planning > execution > demo > inspect and adapt. This steady rhythm is the heartbeat that drives backlog readiness as well. Appearing at a Pre-PI Planning (/pre-and-post-pi-planning/) or a PI Planning (/pi-planning/) without a well-elaborated backlog adds unacceptable risk to the upcoming PI. To this end, the time between PI planning events is a busy time for product and solution management, as they are constantly in the process of refining the backlogs in preparation for the next PI planning. Making this process visible, and achieving "backlog readiness" for the upcoming PI, are the primary purposes of the ART and solution Kanbans. Backlog refinement typically includes:

- Reviewing and updating backlog item definition and developing acceptance criteria
- Working with the teams to establish technical feasibility and scope estimates
- Analyzing ways to split backlog items into smaller chunks of incremental value
- Determining the Enablers (/enablers/) spawned by new features and capabilities, and establishing their capacity allocation

Prioritizing the Backlogs

Prioritizing the backlogs is a key economic driver for the solution. To this end, product and solution management use the Weighted Shortest Job First (/wsjf/) prioritization method for job sequencing. To recap, WSJF ultimately translates to a simple formula:

$$\text{WSJF} = \frac{\text{User | Business Value + Time Criticality + Risk Reduction | Opportunity Enablement Value}}{\text{Job Size}}$$

Applying the formula requires that the numerator items be ranked only on a relative basis to each other. The denominator, job size, can be either a relative measure or an estimate in story points. (Note: The actual denominator of WSJF is job duration, though we use job size as a proxy. This is because duration depends on who does the work and what capacity allocation they can give the new work; that can be almost impossible to determine in advance of allocating the work. Since large jobs typically take longer to do, job size is a reasonable first approximation.)

With respect to job size, the denominator should represent only the *remaining work* for any items already under way. Then, if the remaining work for the item is too large to justify further investment relative to other jobs, the item can be called "good enough," and the teams can move on to other priorities. This implicitly implements Reinertsen's [2] key economic *Principle E17: The sunk cost principle: Do not consider money already spent.*

Preparing for PI Planning

The week or two prior to PI planning is a very busy time. Product and solution management do final backlog preparation, update the vision briefings, and work with Product Owners (/product-owner/) to further socialize the backlog prior to the event. System and Solution Architects/Engineering (/system-and-solution-architect-engineering/) update enabler definitions and models and often develop use cases that illustrate how the features and capabilities work together to deliver the end user value.

Optimizing Value and Solution Integrity with Capacity Allocation

One of the challenges every ART and Solution Train faces is how to balance the backlog of business features and capabilities with the need to continuously invest in the Architectural Runway (/architectural-runway/), provide time for exploration of requirements and design for future PIs, and create prototypes and models to enhance visibility into the problem areas. In order to avoid velocity reduction and to defer the need for wholesale replacement of components due to technological obsolescence, ARTs must invest continuously in implementing the enablers of the solution. This complicates the challenge of prioritizing work since different people can pull the teams in different directions, as Figure 2 shows.

How Much Architecture?

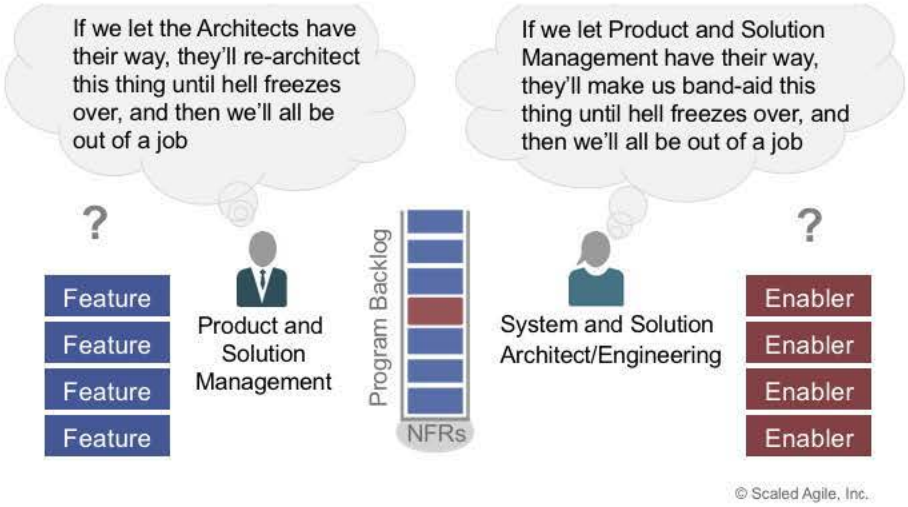


Figure 2. Business vs. enabler backlog items dilemma

To address this problem, teams apply *capacity allocation,* whereby they make a decision about how much of the total effort can be applied for each type of activity for an upcoming PI. Further, they establish an agreement to determine how the work is performed for each activity type. Examples of the results are given in Figure 3 and Table 1.

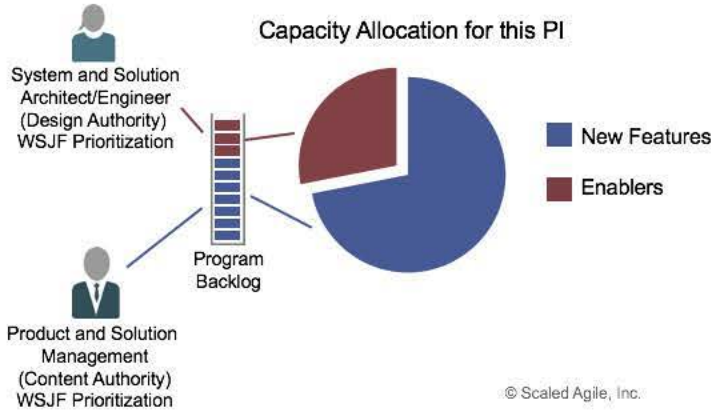


Figure 3. Capacity allocation for architecture in an ART backlog for a single PI

1. At each PI boundary, we agree on the percentage of resources to be devoted to new features or capabilities vs. enablers.
2. We agree that System and Solution Architects and Engineering have authority to prioritize enabler work.
3. We agree that Product and Solution Management have authority to prioritize business backlog items.
4. We agree to jointly prioritize our work based on economics.
5. We agree to collaborate so as to sequence work in a way that maximizes Customer value.

Table 1: Sample policies for managing enabler and feature capacity allocation

While the agreed-to *policies* can persist for some time, the *amount* of capacity allocated should vary over time based on the context. In the context of an ART, this decision can be revisited as part of backlog refinement in preparation for each PI planning, while solution management and solution architect/engineering make similar decisions for the solution as a whole before pre- PI planning.

On Backlogs, Queues, Little's Law, and Wait Times

It's important to take a brief aside and discuss the relationship between backlog, wait times, and flow. The principle *Manage queue length* (/visualize-and-limit-wip-reduce-batch-sizes-and-manage-queue-lengths/) discusses the relationship in detail; refer to that for more complete understanding. However, it's important to summarize that discussion here, because the program and solution backlogs are *the* backlogs that can have the biggest impact on delivery time and throughput. Here's a summary:

1. Little's Law illustrates that the average wait time for an item in a queue is equal to the average length of the queue divided by the average processing rate for an item in a queue. The longer the queue, the higher the wait time, *and* the higher the

variability. (Think of the line at Starbucks: If the ten people ahead of you each order a tall coffee, you are going to be out of there in minutes; if they all order an extra-hot vanilla latte and a heated bagel, you might be late for your meeting, and it is *not* under your control.)

2. Long queues are all bad, causing decreased motivation, poor quality, longer cycle times, higher variability (think Starbucks), and increased risk [2].
3. Your program and solution backlogs are not queues, as items can leapfrog others for faster delivery, and you can always choose not to service everything in the backlog. (Note that neither of these work at Starbucks.)
4. *However*, if all the items in your backlog are committed to stakeholders, *then your backlog behaves like a queue*, and the longer it is, the longer your stakeholders are going to have to wait for service. And if they have to wait too long, they will find another coffee shop, as your shop just can't meet their rapidly changing market needs.

Therefore, in order for a development program to be fast and responsive, *teams must actively manage the backlogs and keep them short*. Teams also must limit commitment to longer-term work, because some other item may come along that's more important than a prior commitment. If a team has too many fixed and committed requirements in the backlog, they cannot respond quickly, no matter how efficient they are. Teams *can be both reliable and fast only if they actively manage the backlog and keep it short*.

Learn More

[1] Leffingwell, Dean. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison-Wesley, 2011.

[2] Reinertsen, Don. *Principles of Product Development Flow: Second Generation Lean Product Development*. Celeritas Publishing, 2009.

Last update: 17 June, 2017

(/)

> *Develop on Cadence. Release on Demand.*
>
> —A SAFe mantra

Release on Demand

Release on Demand is the process by which Features deployed into production are released incrementally or immediately to Customers (/customer/) based on market demand.

Release on demand is the fourth and last element in the four-part Continuous Delivery Pipeline (/continuous-delivery-pipeline/) of Continuous Exploration (/continuous-exploration/) (CE), Continuous Integration (/continuous-integration/) (CI), Continuous Deployment (/continuous-deployment/) (CD), and release on demand, as can be seen in Figure 1.

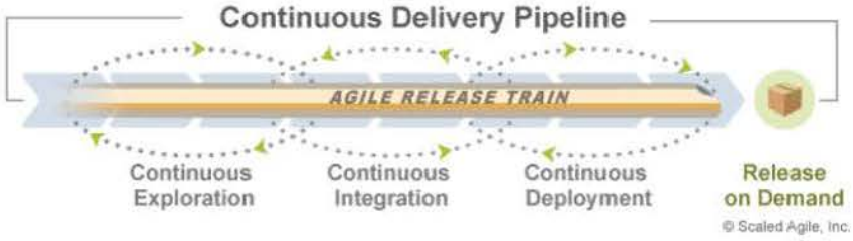


Figure 1. Release on Demand is the final element of the Continuous Delivery Pipeline

The processes that precede release on demand—CE and CI—help ensure that we always have value ready to be deployed in the production environment. But since tangible development value occurs only when end users are successfully operating the Solution (/solution/) in their environment, *releasing* value at the right time is critical for the enterprise to gain the true benefits of agility. This is why the decision to release should never be taken lightly. While there may be situations where deployed features are immediately available to all end users, more often the release is a decoupled on-demand activity, occurring for specific users, timed for when they need it.

Note: This article is the companion article to Develop on Cadence (/develop-on-cadence/).

Details

The ability to release on demand is integral to the continuous delivery pipeline. It preserves three critical options:

- When should we release?
- What elements of the system should be released?
- Which end users should receive the release?

Having all three alternatives accelerates and creates possibility and flexibility in delivering value.

Further, release on demand allows us to 'close the loop' and evaluate the benefit hypothesis proposed during the CE stage. It provides the information needed to decide whether further exploration of an idea is warranted, or if new functionality should be added, or perhaps a different approach is warranted. Achieving the goals of release on demand requires:

- An understanding of how to decouple the release from the development cadence
- An understanding of how to decouple solution elements from a monolithic release
- The ability to architect and re-architect as necessary to support incremental, and partial, delivery

Each of these is described in the following sections.

Decouple on Cadence. Release on Demand

Develop on cadence (/develop-on-cadence/) is the other half of a strategy that allows Agile Release Trains (ARTs) and the Solution Trains to operate in a predictable pattern and synchronize the efforts of multiple development teams. But when it comes to actually releasing value, a different set of rules may apply. Once we have a reliable stream of value through the continuous deployment process, the next and even larger consideration is when and how to actually release all that accumulating value. The release strategy is decoupled from the development cadence, as can be seen in Figure 2.

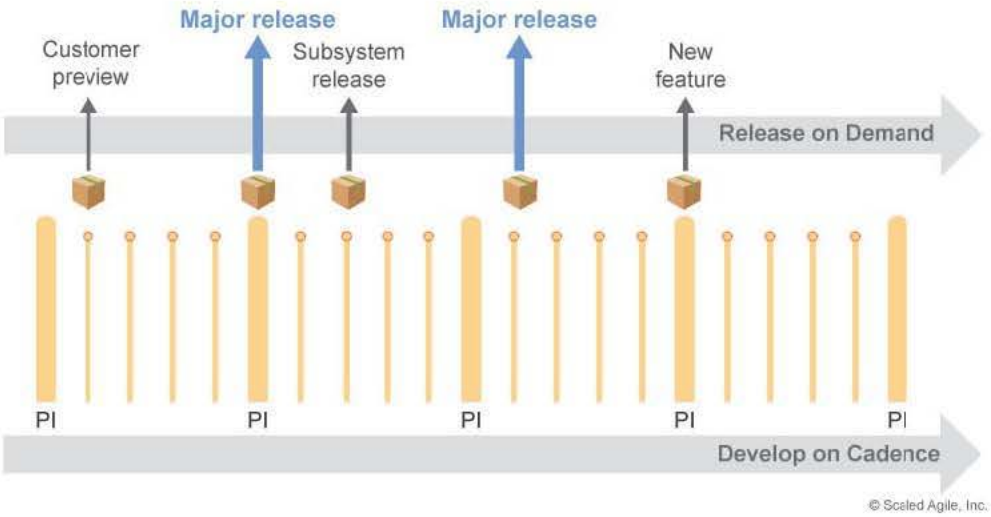


Figure 2. Fully decoupling development concerns from release concerns

There are many release strategies that may apply in different situations:

- **Releasing on the Program Increment (PI) cadence** – The simplest case is when an enterprise can release on the PI (/program-increment/) boundaries. All release dates are known in advance, and PI planning, releasing, and Inspect and Adapt (I&A) (/inspect-and-adapt/) are coordinated by the same cadence and calendar dates. In addition, Innovation and Planning Iterations (/innovation-and-planning-iteration/) can be timed, designed, and coordinated to support the more extensive release activities. This can include final Verification and Validation (V&V), User Accepted Testing (UAT), and release documentation.
- **Releasing less frequently** – In many cases, however, releasing on a fast PI cadence may not be possible or even desirable. For example, in some enterprise settings, deployed systems constitute critical infrastructure for an operating environment. Even if the customer would like to have the new software, the service level and license agreements may be prohibitive. And then there is the overhead and disruption of installation. In other cases, the timelines of enterprises that are building systems with both software and hardware (mobile phones or geophysical mapping satellites) are driven by their long-lead hardware items (displays, chip sets, and the like). The new hardware must be available first, which means releasing early and incrementally is not an option. In these cases, releasing on a PI cadence may not be practical, and the planning and releasing activities may be completely decoupled.
- **Releasing more frequently** – For many, the goal is simply to release as frequently as possible—hourly, daily, weekly, whatever. Achieving that requires the right DevOps (/devops/) capability, an effective continuous delivery pipeline, and an architecture that supports incremental releases. Even then, the periodic planning function still provides the cadence, synchronization, and alignment the enterprise needs to manage variability and limit deviations from expectations.
- **Release on demand** – For enterprises building complex solutions (ex., systems of systems), the cases above are probably overly simplistic. Big systems are not homogeneous. They contain different types of components and subsystems, each of which may leverage its own release model. In that case, the most general model is: Release whatever you want and whenever it makes sense within the governance and business model.

Decouple Release Elements from the Solution

The understanding that we might have different release frequencies also raises another question: Is a release a monolithic, all-or-none process? If so, the release strategy would be limited. Fortunately, that's not the case. In fact, even fairly simple solutions will have multiple release elements, each operating on different release strategies, as Figure 3 illustrates.

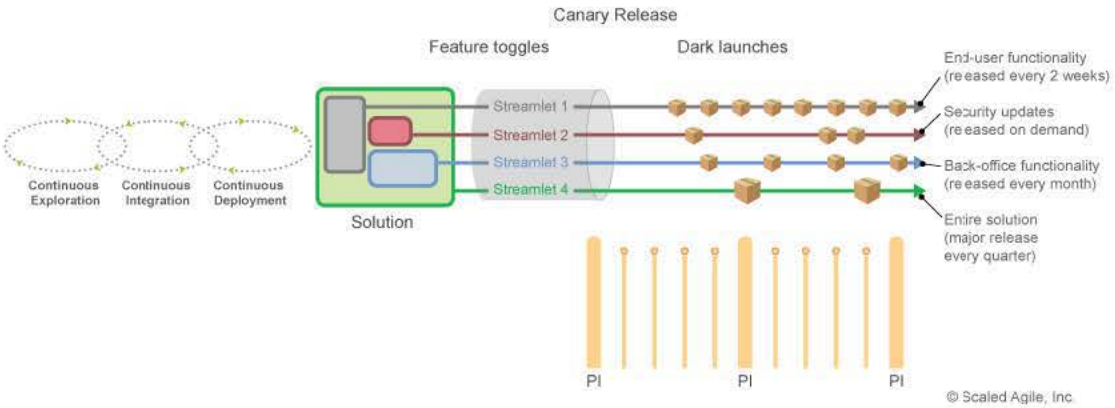


Figure 3. Decouple release elements from the Solution

For example, the SAFe website that's hosting this article has multiple release cycles:

- We can make a fix to deployment or security at any time (ad hoc, but expedited)
- We can update any article at any time and simply notify readers via blog post (high frequency)
- We can add new articles to the Guidance section whenever significant new content is available (medium frequency)
- But we update the major revision to the framework only periodically, based on new content, new ways to render the Big Picture, and most importantly, the readiness of the market for a major new release (low frequency)

We call these separate flows value streamlets, as they represent a full, end-to-end flow of value within the Value Stream. Each streamlet can and should be managed to deliver value according to its own needs and pace. Identifying those streamlets is critical to enable release on demand, as they allow the different elements of the solution to be released independently on their own cadence. They also allow for value stream mapping and improving each streamlet separately.

Architect the Solution for Incremental Release

To achieve these release strategies, and to decouple release elements, the system must be designed for component- or service-based deploy-ability and release-ability, and fast recovery. Designing intentionally to realize these attributes requires the collaboration of System Architects (/system-and-solution-architect-engineering/), Agile Teams (/agile-teams/), and deployment operations. The goal is a fast and flawless delivery process, ideally requiring little manual effort. Capabilities to achieve this more flexible release process include:

- **Feature toggles** – These enable deployment of features into production without making them immediately available to the user. They avoid the need for multiple code branches, allowing developers to deploy new features to production, but to activate them only when the situation warrants. It's important to note that if feature toggles are not cleaned up and removed after they are deemed stable in production, they can become a source of technical debt and reduce the agility of the trains.
- **Canary releases** – Feature toggles also help enable canary releases, which are releases made available only to a subset of users. This allows selective production testing and feature validation without impacting the entire user base.
- **Dark launches** – In some situations, it's important to test new functionality in a production setting before making it available to customers. This is primarily the case where it's uncertain how well the system will meet its Nonfunctional Requirements (NFRs) under production load. Again, feature toggles help manage this risk.

In order to create these capabilities, the organization will typically need to undertake certain enterprise-level architectural initiatives. These may include moving to common technology stacks, decoupling monolithic applications with micro-services, data preparation and normalization, third-party interfaces and data exchange, and logging and reporting tools.

Close the Loop on the Feature Hypothesis

Finally, after the feature is released, it's time to evaluate the benefit hypothesis. Was the intended outcome achieved? For example, should a canary release be extended to more customers? Turn off the feature toggles? Once we have an understanding of the outcome of the release, that feature is done.

Building the Release

Building large-scale systems that are ready for release and fit for use is a process most easily considered in stages, as Figure 4 illustrates.

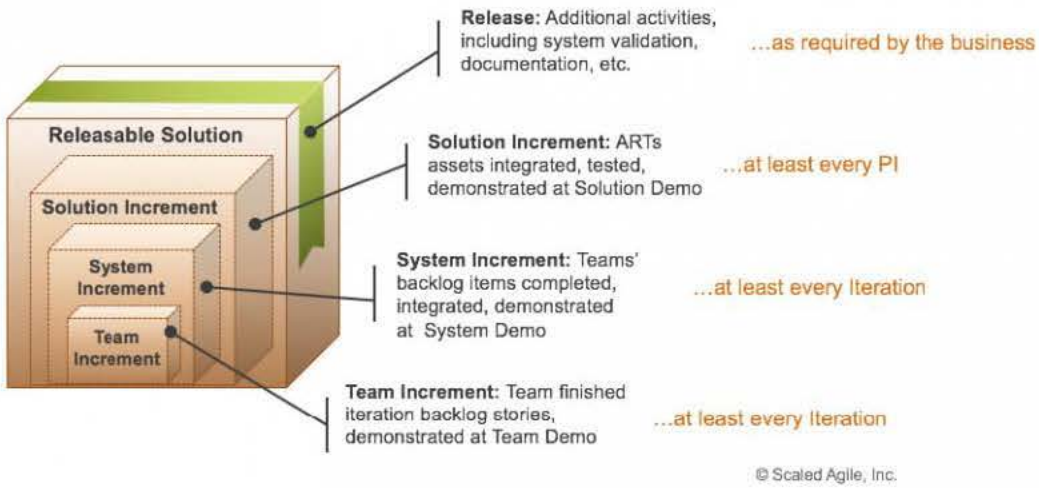


Figure 4. Building a releasable solution

Team Increment

The first, and hopefully, continuous step in this process is that each Agile team ensures that it produces a working increment for the Stories (/story/), features (/features-and-capabilities/), and components they are responsible for. They do this by:

- Completing user stories from their Team Backlog (/team-backlog/), assuring that each meets its story (local) Definition of Done (DoD)
- Applying continuous integration (/continuous-integration/) practices with test automation to monitor and ensure progress

System Increment

Every two weeks, teams build a system increment—an integrated stack of new functionality. This is the sum total of all the backlog items completed by release train teams during the current and all previous iterations (/iterations/). Each system increment is the subject of the System Demo (/system-demo/). In this way, new features (/features-and-capabilities/) are added to the system, one story at a time, typically a few per program increment.

Solution Increment

In large solutions, ARTs typically build only a part of the solution, whether it's a set of subsystems, some solution capabilities, or a mix. For that reason, the real progress of the solution development effort can only be objectively evaluated based on the solution increment that comprises all the capabilities collaboratively delivered by all ARTs in the solution train. Integrated, verified, and validated, they must satisfy both the functional and Nonfunctional Requirements (/nonfunctional-requirements/). This solution increment is the subject of the all-important Solution Demo (/solution-demo/), which brings all the system increments together into a working system.

Release

Building the release in this incremental fashion mainly affects the release of the solution as a whole. As discussed above, many elements can and should be decoupled from the solution to be released independently. These elements might be at the team, system, or solution levels, so the same logic should be applied across the entire integration stack.

Some activities might be required when releasing. These include V&V, release notes, UAT, final documentation, and finalized training materials, as well as marketing activities.

As much as possible, these activities should be included in the DoD of previous increments. Some, however, will remain as release activities.

A Scaled Definition of Done

The continuous buildup of system functionality, along with the final solution itself and the ongoing V&V of its elements, can be reflected in a scaled DoD, as shown below in Table 1.

Team Increment	System Increment	Solution Increment	Release
• Stories satisfy acceptance criteria • Acceptance tests passed (automated where practical) • Unit and component tests coded, passed, and included in the BVT • Cumulative unit tests passed • Assets are under version control • Engineering standards followed • NFRs met • No must-fix defects • Stories accepted by Product Owner	• Stories completed by all teams in the ART and integrated • Completed features meet acceptance criteria • NFRs met • No must-fix defects • Verification and validation of key scenarios • Included in build definition and deployment process • Increment demonstrated, feedback achieved • Accepted by Product Management	• Capabilities completed by all trains and meet acceptance criteria • Deployed/installed in the staging environment • NFRs met • System end-to-end integration, verification, and validation done • No must-fix defects • Included in build definition and deployment/transition process • Documentation updated • Solution demonstrated, feedback achieved • Accepted by Solution Management	• All capabilities done and meet acceptance criteria • End-to-end integration and solution V&V done • Regression testing done • NFRs met • No must-fix defects • Release documentation complete • All standards met • Approved by Solution and Release Management

Table 1. Example SAFe scalable Definition of Done

Release Management

Release management is the process of planning, managing, and governing solution releases, which helps guide the value stream toward the business goals. In some enterprises, especially those with significant regulatory and compliance criteria, this is a centralized, portfolio-level team or function that assures releases meet all the relevant business criteria. In other circumstances, release management and governance are responsibilities shared by the leadership of the ART or solution train, where stakeholders from development operations, quality, sales, and other stakeholders take part in the responsibilities.

In either case, the release management function helps coordinate and facilitate the activities necessary to help internal and external stakeholders receive and deploy the new solution. It also ensures that the most critical governance quality elements have been appropriately addressed prior to deployment —particularly internal and external security, regulatory, and other compliance guidelines.

Release planning happens during PI planning. But planning is the easy part; the hard part is coordinating the implementation of all the capabilities and features over the multiple iterations (/iterations/) within a release. This is especially true when new issues, roadblocks, dependencies, over-scopes, and gaps in Vision (/vision/) and Backlogs (/program-and-solution-backlogs/) are uncovered. This is the challenge for the ARTs; the scope of each release must be continually managed, revalidated, and communicated. Primary considerations include:

- Ensuring that the organization's release governance is understood and adhered to
- Communicating release status to external stakeholders
- Ensuring that an appropriate deployment/distribution plan is in place
- Coordinating with marketing and with Product and Solution Management (/product-and-solution-management/) on internal and external communications
- Validating that the solution meets relevant quality and Compliance (/compliance/) criteria
- Participating in Inspect and Adapt (I&A) (/inspect-and-adapt/) to improve the release process, value stream productivity, and solution quality
- Providing final authorization for the release
- Acting as a liaison with Lean Portfolio Management (LPM) (/lean-portfolio-management/), as appropriate
- Participating in, and overseeing, the final release activities

Many enterprises hold release management meetings weekly to address the following questions:

- Is the vision still understood, and are the trains and teams aligned to that purpose?
- Does everyone understand what they are building, and is it aligned with the understanding of the purpose of the value stream and current Strategic Themes (/strategic-themes/)?
- Are the scheduled releases still largely tracking?
- Is the appropriate quality built in to the solution?
- What impediments must be addressed to facilitate progress?

The weekly meeting provides senior management with regular visibility into the release status. This meeting is also the place to approve any scope, timing, or resource adjustments necessary to ensure the release. In a more continuous delivery environment, the participants monitor the release section of the Program Kanban. Their role is making sure items are

released when needed, to the right audiences, managing dark and canary releases, verifying that hypotheses are evaluated and that feature toggles, etc. are removed after they have been verified.

Learn More

[1] Kim, Gene and Jez Humble, Patrick Debois, John Willis. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press.

Last update: 14 September, 2017

(/)

> *Prediction is very difficult, especially if it is about the future.*
>
> —Niels Bohr

Roadmap

The *Roadmap* is a schedule of events and milestones that communicate planned Solution deliverables over a timeline. It includes commitments for the planned, upcoming Program Increment (PI) and offers visibility into the deliverables forecasted for the next few PIs.

Of course, predicting the future is a hazardous business, and a Lean-Agile Enterprise must be able to respond to changing facts, learning, and business conditions. The real world, however, occasionally demands some certainty. So it may be necessary for enterprises to predict on a longer term basis. (Some initiatives take years to develop, and some degree of commitment must be made to customers, Suppliers, and partners.) To this end, SAFe provides some guidance for forecasting over the longer term, based on the estimated scope of new work, the velocity of the ARTs and Solution Trains, and the current predictability of program execution.

But the desired forecasting horizon must be balanced carefully. Too short, and the enterprise may jeopardize alignment and the ability to communicate important new future Features and Capabilities. Too long, and the enterprise is basing assumptions and commitments on an uncertain future.

Details

"Responding to change over following a plan" is one of the four values of the Agile Manifesto [1]. In order to live up to that value, it should be obvious that it's actually quite important to *have* a plan, as otherwise everything is a change, and the backlog is the "tail of the dog that is constantly wagged" by changes that should have been readily anticipated. In turn, this causes thrashing, excess rework, and excess WIP. It is demotivating to all. Planning is good.

The *Roadmap* provides just such a plan. In the context of a roadmap, the teams know what their current commitments are and what the plan of intent is. The ability to routinely execute on those planned tasks provides a sense of personal satisfaction, as well as the extra mental and physical capacity necessary to *respond to the real* changes, those that could not have been anticipated.

The SAFe roadmap consists of a series of planned Program Increments (PIs) (/program-increment/) with various Milestones (/milestones/) called out, as is indicated by Figure 1.

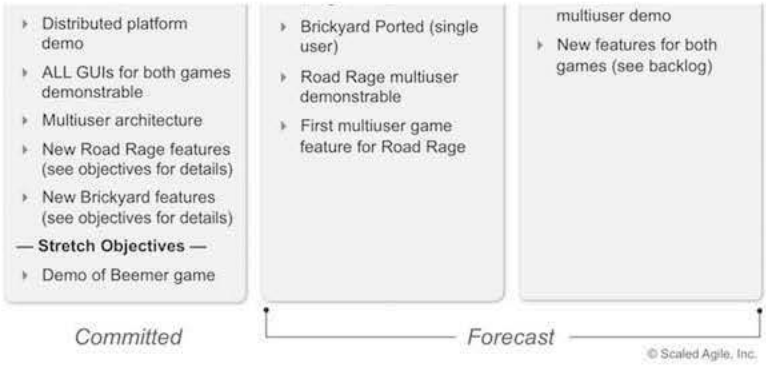


Figure 1. An example PI Roadmap for a gaming company

Each element on the roadmap is a milestone, either a learning milestone that has been defined by the teams or a date milestone that may be driven by external events.

Building the PI Roadmap

The Figure 1 roadmap covers three PIs, or approximately 30 weeks. In many Enterprises (/enterprise/), this is about the sweet spot—there's enough detail to be able to run the business, and yet a short enough time frame to keep long-term commitments from interfering with the ability to flex to changing business priorities. This roadmap consists of a "committed PI" and two "forecasted PIs," as described in the following sections.

The Committed PI

During PI Planning (/pi-planning/), teams commit to meeting the program PI Objectives (/pi-objectives/) for the upcoming PI. Therefore the near-term plan is a high-confidence plan; the enterprise should be able to confidently plan for the impact of the upcoming new functionality. For Agile Release Trains (ARTs) (/agile-release-train/) that are new to SAFe and have yet to reach high confidence levels with their PI plans, the System Demo (/system-demo/) and Inspect and Adapt (/inspect-and-adapt/) workshop will help increase confidence in each upcoming PI. In any case, reaching a predictable delivery of the upcoming PI is an importable capability for every ART.

Forecast PIs

Forecasting the next two PIs is a little more interesting. ARTs and Solution Trains (/solution-train/) typically plan only one PI at a time. For most, it's simply imprudent to plan in detail much further (except perhaps for some Architectural Runway (/architectural-runway/)), because the business or technical context changes so quickly.

However, the Program and Solution Backlogs (/program-and-solution-backlogs/) contain Featured and Capabilities (/features-and-capabilities/) (which constitute future milestones) that have been working their way through the Kanban (/program-and-solution-kanbans/) systems. They've been reasoned, socialized with the teams, have acceptance criteria in process, and have preliminary estimates for size in Story (/story/) points. Given knowledge of the ART velocities, the PI predictability Measure (/metrics/), relative priorities, and the history of how much work is devoted to maintenance and other business-as-usual activities, ARTs can generally lay the future features into the roadmap without too much difficulty. The result is that most trains have roadmaps with a reasonable degree of confidence over about a three-PI period.

Long-Term Forecasting

The above discussion highlights how enterprises can have a reasonable, near-term plan of intent for all the ARTs in the portfolio. However, for many enterprises, especially those building large, complex systems, complete with Suppliers (/supplier/), long hardware lead times, major subsystems, critical delivery dates, external Customer (/customer/) commitments, etc., that amount of roadmap will be inadequate. Simply, building a satellite, a crop combine, or a new car takes a lot longer than the PI roadmap, and the enterprise must plan realistically for the longer-term future.

In addition, even when external events do not necessarily drive the requirement for long-term forecasting, enterprises need to be able to plan investment in future periods. They need to understand the potential recourse and development bottlenecks in support of the longer-term business demands, and the waxing and waning of investment in particular value streams.

So the conclusion is inevitable: it is most likely necessary to extend the forecast well beyond the PI planning horizon, even though the future work is largely unplanned.

Estimating Longer-Term Initiatives

Estimating Longer-Term Initiatives

Fortunately, Agile work physics give us a means to forecast longer-term work. Of course, in order to forecast the work, estimation is required. In order to estimate, teams can use Agile story point physics, based on normalized estimating, and estimate larger initiatives at the Epic (/epic/) level, as Figure 2 illustrates.

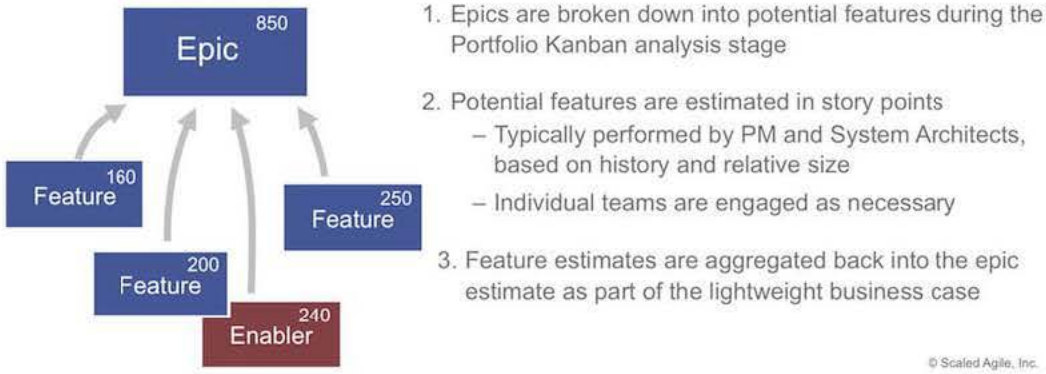


Figure 2. Epics are estimated in story points rolled up from feature estimates

Given these estimates, a knowledge of ART velocities, and some sense of the capacity allocation that can be provided to the new initiative, the business can then play out a "what-if" analysis, as Figure 3 illustrates.

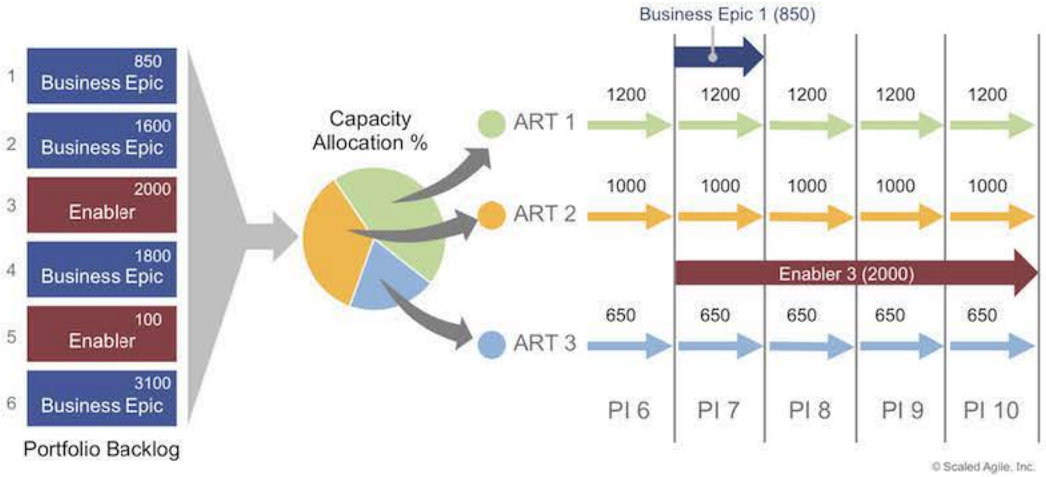


Figure 3. Epic estimates, capacity allocations, and ART velocities are used for longer-term forecasting

Therefore, the enterprise has a way to build a roadmap that goes as far into the future as they need. However, every enterprise must be very careful about such forecasts, and while many see long-term predictability as the goal, Lean-Agile Leaders (/lean-agile-leaders/) know that *every long-term commitment decreases the agility of the enterprise*. Even in a case in which requirements can be fairly well established in advance, the unpredictability and variability of innovation, the tendency to estimate only the known activities, the difficulty in predicting future capacity, unforeseen events, and other factors all conspire to create a bias for underestimating reality. The net effect is that while fixed, long-term plans and commitments may feel good, and may even be required in some circumstances, they absolutely limit the ability of the enterprise to pivot to a new, and potentially more economically beneficial, outcome. You can't have it both ways.

As a result, the Lean-Agile enterprise strives to establish the right amount of visibility, alignment, and commitment, while enabling the right amount of flexibility. The correct balance can be obtained via a willingness to convert long-term *commitment* to *forecasts* and to continually reevaluate priorities and business value, as needs dictate, at each PI boundary.

Avoid the Queue from Hell

Lean-Agile Leaders must understand the queuing theory discussed in SAFe's principle #6 (visualize and limit WIP, reduce batch sizes, and manage queue lengths (/visualize-and-limit-wip-reduce-batch-sizes-and-manage-queue-lengths/)) and be aware of the impact that long queues have on delivery time. Simply, the longer the committed queue, the longer the wait for any new initiative. Period. For example, in Figure 1 above, the second PI on the roadmap does not appear to be fully loaded. The math then tells us that the wait time for a new, unplanned feature is from 10 to 20 weeks; it can't be in the the current PI, but it can be in the next.

The roadmap in Figure 4, below tells a different story.

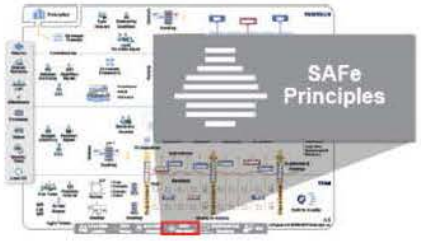
Figure 4. A fully committed roadmap is the queue from hell

If, for example, all the items on this roadmap are fully committed and the teams are running at nearly full capacity, then the wait time for a new capability is in excess of 60 weeks! The enterprise while thinking it's agile, is really stuck in a traditional mindset. If they do not understand Little's Law (https://www.youtube.com/watch?v=IHQZcMRr2n0) of queueing theory.

Learn More

[1] http://agilemanifesto.org (http://agilemanifesto.org)

Last updated: 19 July, 2017



(/)

> *The impression that "our problems are different" is a common disease that afflicts management the world over. They are different, to be sure, but the principles that will help to improve the quality of product and service are universal in nature.*
> *—W. Edwards Deming*

SAFe Principles

SAFe is based on nine immutable, underlying *Lean and Agile Principles*. These are the fundamental tenets, basic truths, and economic premises that inspire and inform the roles and practices that make SAFe effective:

#1-Take an economic view

#2-Apply systems thinking

#3-Assume variability; preserve options

#4-Build incrementally with fast, integrated learning cycles

#5-Base milestones on objective evaluation of working systems

#6-Visualize and limit WIP, reduce batch sizes, and manage queue lengths

#7-Apply cadence, synchronize with cross-domain planning

#8-Unlock the intrinsic motivation of knowledge workers

#9-Decentralize decision-making

Why the Focus on Principles?

Building enterprise-class software and cyber-physical systems is one of the most complex challenges the industry faces today. Millions of lines of software, complex hardware and software interactions, multiple concurrent platforms, demanding and unforgiving nonfunctional requirements—these are just a few of the challenges systems builders face.

Of course, the enterprises that build these systems are increasingly complex, too. They are bigger and more distributed than ever. Mergers and acquisitions, distributed multinational (and multilingual) development, offshoring, and the rapid growth that success requires are all part of the solution—but also part of the problem as well.

Fortunately, we have an amazing and growing body of knowledge to help us address this challenge. These include Agile principles and methods, Lean and Systems thinking, product development flow, Lean process and product development, and more. Many thought leaders have gone down this path before us and left a trail to follow in the hundreds of books and references we can draw on.

SAFe's goal is to synthesize some of this body of knowledge and the lessons learned from hundreds of deployments into a single framework—a system of integrated, proven practices that has been demonstrated to bring substantial improvements in employee engagement, time to market, solution quality, and team productivity. However, given the complexity of the

employee engagement, time to market, solution quality, and team productivity. However, given the complexity of the industry challenges already discussed, there is truly no off-the-shelf solution to the unique challenges every enterprise faces. This means that some tailoring and customization may be required, as not every SAFe-recommended practice will apply equally well in every circumstance. Therefore, we always endeavor to make certain that SAFe practices are grounded in fundamental, and reasonably immutable, principles. In that way, we can be confident that they apply well in the general case. And when and if they don't, the underlying principles can guide those doing the implementation to make sure that they are moving on a continuous path to the "shortest sustainable lead time, with best quality and value to people and society." There is value in that too.

SAFe's practices are grounded on nine fundamental principles that have evolved from Agile principles and methods, Lean product development, systems thinking, and observation of successful enterprises. Each principle is described in detail in an article of that name. In addition, the embodiment of the principles appears throughout the framework. As a further aid to understanding, they are summarized below.

#1 – Take an economic view (/take-an-economic-view/)

Achieving the best value and quality to people and society in the sustainably shortest lead time requires a fundamental understanding of the economics of the system builder's mission. Lean systems builders endeavor to make sure that every day 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 value stream economic framework, which defines the tradeoffs between risk, cost of delay, operational and development costs, and supports decentralized decision-making.

#2- Apply systems thinking (/apply-systems-thinking/)

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 systems of all types—manufacturing systems, social systems, management systems, even government systems. One central conclusion was the understating 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 do their work. In other words, it's a s*ystem problem* not a *people problem*, and that requires systems thinking. This principle describes how systems thinking can be applied to the system you are building, the organization that builds the system, and to the entire value stream that produces the system.

#3 – Assume variability; preserve options (/assume-variability-preserve-options/)

Traditional design and lifecycle practices drive picking a single requirements and design option early in the development process (early in the "cone of uncertainty). However, if the starting point is wrong, then future adjustments take too long and can lead to a suboptimal long-term design. Alternatively, lean systems developers 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.

#4 – Build incrementally with fast, integrated learning cycles (/build-incrementally-with-fast-integrated-learning-cycles/)

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

#5 – Base milestones on objective evaluation of working systems (/base-milestones-on-objective-evaluation-of-working-systems/)

Systems builders 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 opportunity 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 a continuing investment will produce a commensurate return.

#6 – Visualize and limit WIP, reduce batch sizes, and manage queue lengths (/visualize-and-limit-wip-reduce-batch-sizes-and-manage-queue-lengths/)

Lean systems builders strive to achieve a state of continuous flow, whereby new system capabilities move quickly and visibly from concept to cash. Three primary keys to implementing flow are to: 1. Visualize and limit the amount of work-in-process so as to limit demand to actual capacity, 2. Reduce the batch sizes of work items to facilitate reliable flow through the system, and 3. Manage queue lengths so as to reduce the wait times for new capabilities.

#7 – Apply cadence, synchronize with cross-domain planning (/apply-cadence-synchronize-with

#7 – Apply cadence, synchronize with cross-domain planning (/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, provides Lean systems builders with the tools they need to operate effectively in the presence of product development uncertainty.

#8 – Unlock the intrinsic motivation of knowledge workers (/unlock-the-intrinsic-motivation-of-knowledge-workers/)

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

#9 – Decentralize decision-making (/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 fidelity decisions, due to the lack of local context, plus changes in fact patterns that occur during the wait time. Decentralized decision-making reduces delays, improves product development flow and enables faster feedback and more innovative solutions. However, some decisions are strategic, global in nature, and have economies of scale sufficient enough to warrant centralize decision-making. Since both types of decisions occur, the creation of an effective decision-making framework is a critical step in ensuring fast flow of value.

Last update: 24 August, 2016

(/)

> *Stories act as a "pidgin language," where both sides (users and developers) can agree enough to work together effectively.*
>
> —Bill Wake, co-inventor of Extreme Programming

Stories

Stories are short descriptions of a small piece of desired functionality, written in the user's language. Stories implement small, vertical slices of system functionality, sized so they can be completed by the Agile Team in a single Iteration.

Stories are the primary artifact used to define system behavior in Agile. They're not requirements. Rather, they're short, simple descriptions of a small piece of desired functionality, usually told from the user's perspective and written in their language. Each one is intended to enable the implementation of a small, vertical slice of system functionality, supporting highly incremental development.

Stories provide just enough information for the intent to be understood by both business and technical people. Details are deferred until the story is ready to be implemented. Through acceptance criteria, stories get more specific as they are implemented, helping ensure system quality.

User stories deliver functionality directly to the end user. *Enabler stories* bring visibility to the work items needed to support exploration, architecture, and infrastructure.

Details

SAFe describes a four-tier hierarchy of artifacts that describe functional system behavior: Epic (/epic/)> Capability (/features-and-capabilities/) > Feature (/features-and-capabilities/) > *Story*. These, along with Nonfunctional Requirements (NFRs) (/nonfunctional-requirements/), are the Agile requirements (system behavioral) artifacts that are used to define system and Solution Intent (/solution-intent/), model system behavior, and build up the Architectural Runway (/architectural-runway/).

Epics, capabilities, features, and Enablers (/enablers/) are used to describe the larger intended behavior, but the detailed implementation work is described via *stories*, which constitute the Team Backlog (/team-backlog/). Most stories arise from business and enabler features in the Program Backlog (/program-and-solution-backlogs/), but others emerge from the team's local context.

Each story is a small, independent behavior that can be implemented incrementally and that provides some value to the user or the Solution (/solution/); it is a vertical slice of functionality to help ensure that every Iteration (/iterations/) delivers new value. To accomplish this, stories are split (see below) as necessary so they can be completed in a single Iteration.

Initially, stories are typically written on an index card or sticky note. The physical nature of the card creates a tangible relationship between the team, the story, and the user and helps engage the entire team in story writing. They have a kinesthetic element as well; they help visualize work (/visualize-and-limit-wip-reduce-batch-sizes-and-manage-queue-lengths/) and can be readily placed on a wall or table, rearranged in sequence, passed around, and even handed off when necessary. They help teams better understand scope ("Wow, look at all these stories I'm about to sign up for") and progress ("Look at all the stories we accomplished in this Iteration").

While anyone can write stories, approving stories into the team backlog and accepting them into the system baseline is the responsibility of the Product Owner. Of course, stickies don't scale well across the Enterprise (/enterprise/), so stories often

responsibility of the Product Owner. Of course, stickies don't scale well across the enterprise (enterprise), so stories often move quickly into Agile project management tooling.

There are two types of stories in SAFe, user stories and enabler stories, as described below.

Sources of Stories

In SAFe, stories are generally driven by splitting business features and enabler features, as Figure 1 illustrates.

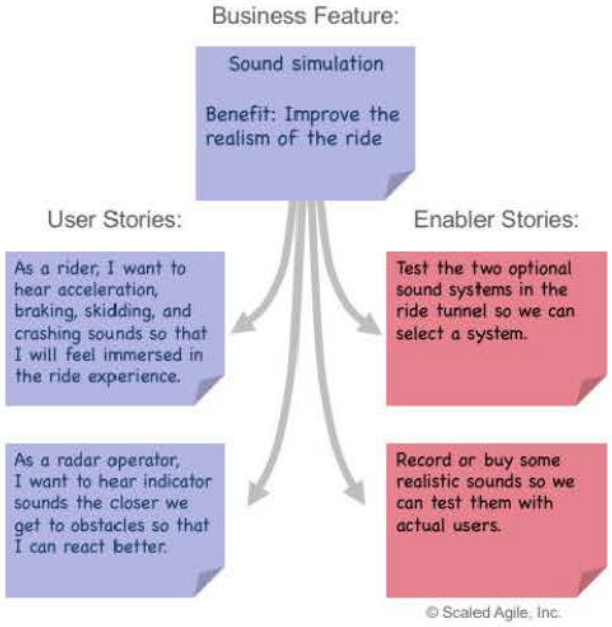


Figure 1. Example of a business feature split into stories

User Stories

User stories are the primary means of expressing needed functionality. They largely replace the traditional requirements specification. (In some cases, however, they serve to understand and develop functionality that is later recorded in such a document in support of compliance, traceability, or other needs.)

User stories are *value centric* in that they focus on the user, not the system, as the subject of interest. In support of this, the recommended form of expression is the "user voice form," as follows:
As a 〈user role〉 I want 〈activity〉 so that 〈business value〉

By using this format, the teams are constantly guided to understand *who* is using the system, *what* specifically they are doing with it, and *why* they are doing it. Applying the user voice routinely tends to increase the team's domain competence; they come to better understand the real business needs of their user. Figure 2 provides an example:

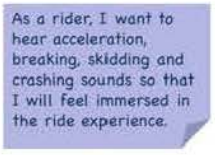


Figure 2. Example user story in user voice form

While the user story voice is the common case, not every system interacts with an end user. Sometimes the "user" is a *device* (example: printer) or other *system* (example: transaction server). In this case, the story can take on the form illustrated in Figure 3.

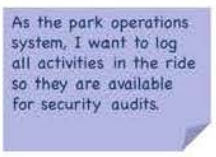


Figure 3. Example of a user story with a system as a user

Enabler Stories

Teams also need to develop technical functionality that is needed to implement a number of different user stories, or support other components of the system. In this case, the story may not directly touch any end user. These are *enabler stories*, and they can support *exploration*, *architecture*, or *infrastructure*, just like all other enablers. In these cases, the story can be expressed in technical rather than user-centric language, as Figure 4 illustrates.

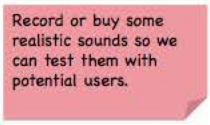


Figure 4. Example enabler story

Enabler stories may include any of the following:

- Refactoring (/refactoring/) and Spikes (/spikes/) (as traditionally defined in XP)
- Building or improving development/deployment infrastructure
- Running jobs that require human interaction (example: *Index 1 million web pages*)
- Creating required product or component configurations for different purposes
- Performing special types of system qualities verification (vulnerability testing, etc.)

And, of course, enabler stories are demonstrated just like user stories, typically via showing the artifacts produced or via UI, stub, or mock.

Writing Good Stories

The 3Cs: Card, Conversation, Confirmation

Ron Jeffries, one of the inventors of XP, is credited with describing the "3Cs" of a story:

Card represents the capture of the *statement of intent* of the user story on an index card, sticky note, or tool. The use of index cards provides a physical relationship between the team and the story. The card size physically limits the length of the story and, thereby, too-early specificity of system behavior. Cards also help the team "feel" upcoming scope, as there is something materially different about holding 10 cards in one's hand versus looking at 10 lines on a spreadsheet.

Conversation represents a "promise for a conversation" between the team, Customer (/customer/)/user, the Product Owner (/product-owner/), and other stakeholders. This is the discussion necessary to determine the more detailed behavior required to implement the intent. The conversation may spawn additional specificity in the form of attachments to the user story (mock-up, prototype, spreadsheet, algorithm, timing diagram, etc). The *conversation* spans all steps in the story life cycle:

- Backlog refinement
- Planning
- Implementation
- Demonstration

Conversations provide shared context that cannot be achieved via formal documentation. It drives away requirements ambiguity via concrete examples of functionality. The conversation helps uncover gaps in scenarios and nonfunctional requirements. Some teams also use the confirmation section of the story card to write down what they will demo for the story.

Confirmation the *acceptance criteria* and provides the precision necessary to ensure that the story is implemented correctly and covers the relevant functional and nonfunctional requirements. Figure 5 provides an example.

Figure 5. Story acceptance criteria

Agile Teams (/agile-teams/) automate acceptance tests wherever possible, often in a business-readable, domain-specific language, thereby creating the "automatically executable specification and test" of the code. Automation also provides the ability to quickly regression-test the system, which enhances Continuous Integration (/continuous-integration/), refactoring, and maintenance.

Invest in Good Stories

People often use the mnemonic INVEST, (http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/) developed by Bill Wake (https://www.industriallogic.com/people/Bill), to provide a reminder of the things that make a good story:

- **I** – Independent (of all other stories)
- **N** – Negotiable (a flexible statement of intent, not a contract)
- **V** – Valuable (providing a valuable vertical slice to the Customer)
- **E** – Estimable (small and negotiable)
- **S** – Small (fits within an iteration)
- **T** – Testable (understood enough to know how to test it)

Refer to [1] and [2] for more information.

Estimating Stories

Agile Teams use story points and estimating poker [2 and 3] to estimate their work. A story point is a *singular* number that represents a combination of things:

- Volume – How much is there?
- Complexity – How hard is it?
- Knowledge – What's known?
- Uncertainty – What's not known?

Story points are relative; they are *not* connected to any specific unit of measure. The size (effort) of each story is estimated relative to the smallest story, which is arbitrarily assigned a size of 1. SAFe applies the modified Fibonacci sequence (1, 2, 3, 5, 8, 13, 20, 40, 100) to reflect the inherent uncertainty in estimating, especially large numbers (e.g. 20, 40, 100, etc.) [2].

Estimating Poker

Agile Teams often use "estimating poker," which combines expert opinion, analogy, and disaggregation for quick but reliable estimates (note that there are a number of other methods used as well). The rules of estimating poker are:

- Participants include all team members
- Each estimator is given a deck of cards with 1, 2, 3, 5, 8, 13, 20, 40, 100, ∞, and ?
- The Product Owner participates but does not estimate
- The Scrum Master (/scrum-master/) participates but does not estimate; an exception is if he or she is doing actual development work
- For each backlog item to be estimated, the Product Owner reads the description of the story
- Questions are asked and answered
- Each estimator privately selects an estimating card representing his or her estimate
- All cards are simultaneously turned over so that all participants can see each estimate
- High and low estimators explain their estimates
- After discussion, each estimator reestimates by selecting a card
- The estimates will likely converge; if not, repeat the process

Some amount of preliminary design discussion is appropriate. However, spending too much time on design discussions is often wasted effort. The real value of estimation poker is to come to a common agreement on the scope of a story. And it's also fun!

Velocity

The team's *velocity* for an iteration is equal to the sum of the points for all the stories completed (that have met their Definition of Done) in the iteration. Knowing velocity assists with planning and is a key factor in limiting WIP, as 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 larger epics, features, capabilities, and enablers, which are also estimated in story points.

Common Starting Baseline for Estimation

In standard Scrum, each team's story point estimating—and the resultant velocity—is a local and independent concern; the fact that a small team might estimate in such a way that they have a velocity of 50 while a larger team has a velocity of 12 is of no concern to anyone.

In SAFe, however, story point velocity must have a common starting baseline, so that estimates for features or epics that require the support of many teams are based on rational economics. In order to achieve this, SAFe teams start down a path on which a story point for one team is roughly the same as a story point for another, so that, with adjustments for economics of location (U.S., Europe, India, China, etc.), work can be estimated and prioritized based on converting story points to cost. After all, there is no way to determine the return on potential investment if there is no comparable "currency."

The method for getting to a common starting baseline for *stories* and *velocity* is as follows:

- For every developer-tester on the team, give the team eight points (adjust for part-timers).
- Subtract one point for every team member vacation day and holiday.
- Find a small story that would take about a half-day to code and a half-day to test and validate. Call it a one (1).
- Estimate every other story relative to that one (1).

Example: Assuming a six (6)-person team composed of three (3) developers, two (2) testers, and one PO, with no vacations or holidays, the estimated initial velocity = 5 * 8 points = 40 points/iteration. (Note: Adjusting a bit lower may be necessary if one of the developers and testers is also the Scrum Master.)

In this way, story points are somewhat comparable to an ideal developer day, and all teams estimate size of work in a common fashion, so management can thereby fairly quickly estimate the cost for a story point for teams in a specific region. Then they have a meaningful way to figure out the cost estimate for an upcoming feature or epic.

Note: There is no need to recalibrate team estimation or velocity after that point. It is just a common starting baseline.

While teams will tend to increase their velocity over time—and that is a good thing—in fact the number tends to remain fairly stable. A team's velocity is far more affected by changing team size and technical context than by productivity changes. And if necessary, financial planners can adjust the cost per story point a bit. Experience shows that this is a minor concern, compared to the wildly differing velocities that teams of comparable size may have if they don't set a common starting baseline. That simply doesn't work at enterprise scale, because decisions can't be based on economics that way.

Splitting Stories

Smaller stories allow for faster, more reliable implementation, since small things go through a system faster, reducing variability and managing risk. Splitting bigger stories into smaller ones is thus a mandatory survival skill for every Agile Team, and it is both the art and the science of incremental development. Ten ways to split stories are highlighted in [1]. A summary of these techniques is included below.

1. Work flow steps
2. Business rule variations
3. Major effort
4. Simple/complex
5. Variations in data
6. Data entry methods
7. Deferred system qualities
8. Operations (example: Create Read Update Delete, or CRUD)
9. Use-case scenarios
10. Break-out spike

Figure 6 illustrates an example of #9, splitting by use-case scenarios.

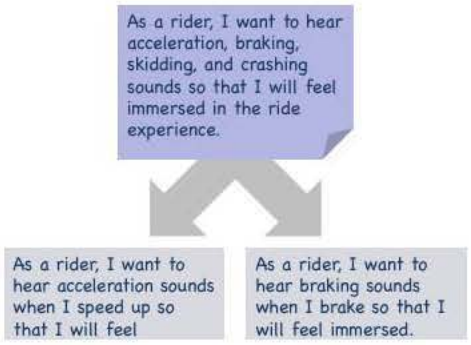


Figure 6. An example of splitting a large story into smaller stories

Stories in the SAFe Requirements Model

As described in the SAFe Requirements Model (/safe-requirements-model/), the framework applies an extensive set of artifacts and relationships to manage the definition and testing of complex systems in a Lean and Agile fashion. Figure 7 illustrates the role of stories in this larger picture.

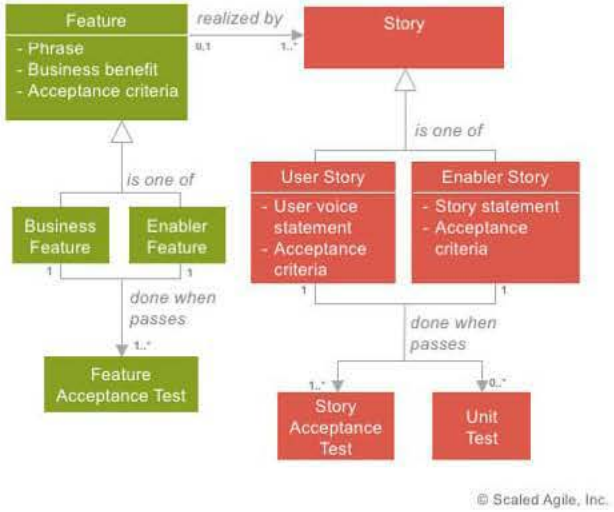


Figure 7. Stories in the SAFe Requirements Model

Figure 7 illustrates how stories are often, but not always, spawned by new features, and how each has an associated story acceptance test. Further, in XP and SAFe ScrumXP, each story should have a unit test associated with it. The unit tests serves primarily to ensure that implementation of the story is correct. In addition, as unit tests are readily able to be automated, this is a critical starting point for test automation, as described in Test-First (/test-first/).

Note: Figure 7. uses UML notation to represent the relationships between the objects: zero to many (0,1), one to many (1..*), one to one (1) and so on.

Learn More

[1] Leffingwell, Dean. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison-Wesley, 2011.

[2] Cohn, Mike. *User Stories Applied: For Agile Software Development*. Addison-Wesley, 2004.

Last update: 21 June, 2017

Latest Updates (/blog/)

TRAINING

Course Calendar (http://www.scaledagileacademy.com/events/event_list.asp)

About Certification (http://www.scaledagileacademy.com/?page=WhichCertification)

Become a Trainer (http://www.scaledagileacademy.com/?page=becomeatrainer)

PERMISSIONS

Permission FAQ (http://www.scaledagile.com/permission-faq/)

Permissions Form (http://www.scaledagile.com/permissions-form/)

Usage and Permissions (/usage-and-permissions/)

PARTNER

Becoming a Partner (http://www.scaledagile.com/become-a-partner/)

Partner Directory (http://www.scaledagile.com/listingcategory/directory/)

Partner Event Calendar (http://www.scaledagile.com/event-list/)

GET SOCIAL

Twitter (https://twitter.com/ScaledAgile)

Linkedin (https://www.linkedin.com/grps/Scaled-Agile-Framework-4189072?)

YouTube (https://www.youtube.com/user/scaledagile)

SlideShare (http://www.slideshare.net/ScaledAgile)

RECENT POSTS

Updated SAFe Scrum Master and SAFe Advanced Scrum Master courses now available! (/blog/updated-safe-scrum-master-and-safe-advanced-scrum-master-courses-now-available/)

Sep, 19th 2017

SAFe from the Trenches at Agile Australia (/blog/safe-from-the-trenches-at-agile-australia/)

Sep, 19th 2017

Updated SAFe Release Train Engineer and SAFe Product Owner/Product Manager courses now available! (/blog/updated-safe-release-train-engineer-and-safe-product-ownerproduct-manager-courses-now-available/)

Aug, 31th 2017

SCALED AGILE, INC

CONTACT US

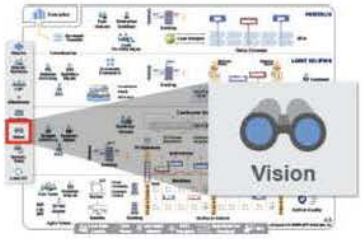
5480 Valmont Rd., Suite 100
Boulder, CO 80301 USA

Email: support@scaledagile.com (mailto:support@scaledagile.com)

Phone: +1.303.554-4367

BUSINESS HOURS

Weekdays: 9am to 5pm
Weekends: CLOSED



(/)

> *People at work are thirsting for context, yearning to know that what they do contributes to a larger whole.*
>
> —Daniel Pink

Vision

The *Vision* is a description of the future state of the Solution under development. It reflects customer and stakeholder needs, as well as the features and capabilities proposed to meet those needs.

The vision is both aspirational and achievable, providing the larger context—an overview and purpose—of the solution under development. It describes the markets, customer segments, and user needs to be served. It sets the boundaries against which new features, Nonfunctional Requirements (NFRs), and other content decisions are based.

The vision icon lives on the 'spanning palette,' which means it can apply to any level of SAFe. While the focus of the vision is typically is on the solution, a Portfolio Vision is also clearly relevant, reflecting how the Value Streams (/value-streams/) will cooperate to achieve the larger Enterprise (/enterprise/) objectives. Moreover, Agile teams will often also have a vision for the elements of the system for which they have responsibility.

Details

Few question the benefit of Lean-Agile's focus on near-term deliverables and fast value delivery, which favors: deferring decisions until the *last responsible moment*, limiting work in process, avoiding Big Upfront Design (BUFD) and future proofing architectures, along with overly detailed long-term plans. There is no good substitute for a bias for action: 'Let's build it and then we'll know.'

However, in the context of large Solutions (/solution/), every individual contributor makes frequent decisions on what the solution should do do now and, in so doing, also what will be easy—or not so easy—to do in the future. In this way, they ultimately determine both the effectiveness of the current solution and the economics of developing future solutions.

Continuously developing, maintaining, and communicating the vision—both near and longer term—is critical to creating a shared understanding of the program's goals and objectives, especially as those ideas evolve due to ever-shifting market needs and business drivers.

Portfolio Vision

The portfolio vision sets a longer-term context for near-term decisions in a way that is both practical and inspirational: "This is something really worth doing." Local decisions by Agile teams are best made in the light of this longer-term context.

Lean-Agile leaders have the most responsibility for setting the strategic direction of the company and establishing the mission for the teams who will implement that strategy. *Switch* [1] calls this longer-term view a "Destination Postcard," as Figure 1 illustrates.

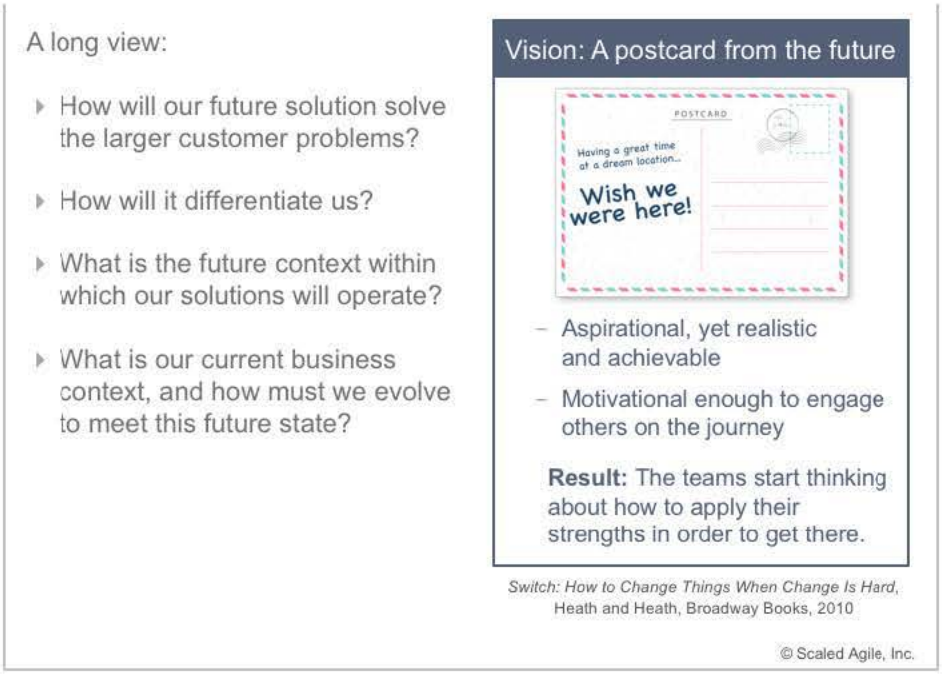


Figure 1. The portfolio vision is an enterprise-level "postcard from the future"

A portfolio vision exhibits the following characteristics:

- **Aspirational, yet realistic and achievable.** The vision has to be compelling and somewhat futuristic, yet realistic enough to be achievable over some meaningful timeframe.
- **Motivational enough to engage others on the journey.** The vision must align with the Strategic Themes (/strategic-themes/), as well as to the individual team's purpose.

This longer-term vision—and the business context that drives it—is typically delivered to the teams during the PI Planning (/pi-planning/) event. It may be delivered by the Business Owners (/business-owners/), or possibly by the C-level executive who can best communicate the vision, as well as inspire others to help achieve it. Given this vision, the teams will start thinking about how they will apply their unique strengths. True engagement begins with a destination in mind, one that helps the teams fulfill their purpose.

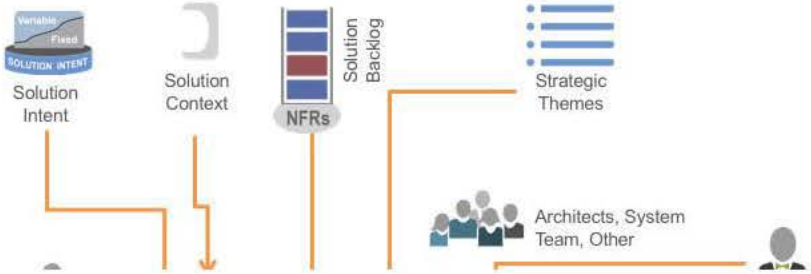
Solution Vision

Given that longer-term view, Product and Solution Management (/product-and-solution-management/) have the responsibility for converting the portfolio vision to a solution vision indicating the reason and direction behind the chosen solution. In order to do so, specific questions must be asked and answered:

- What will this new solution do?
- What problem will it solve?
- What Features (/features-and-capabilities/) and *benefits* will it provide?
- For whom will it provide them?
- What performance (Nonfunctional Requirements (/nonfunctional-requirements/)) will it deliver?

Inputs to the Solution Vision

In SAFe, this responsibility lies primarily with Product and Solution Management. They work directly with the Business Owners and other stakeholders to synthesize all the inputs and integrate them into a holistic and cohesive vision. There is no shortage of inputs, as Figure 2 illustrates.



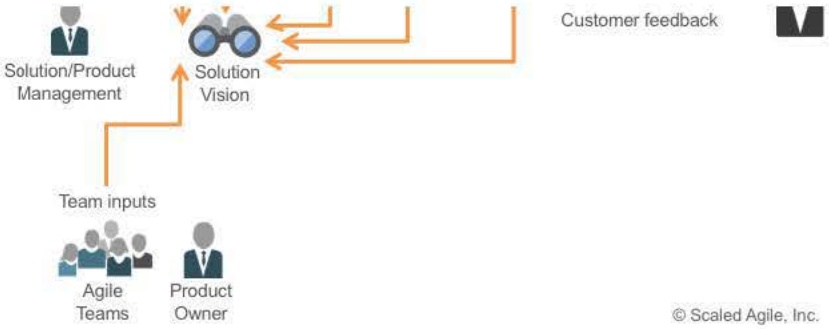


Figure 2. Solution vision input sources

These inputs include:

- Customers (/customer/) provide fast feedback and have intimate knowledge of what is needed
- Solution Intent (/solution-intent/) contains some of the vision and is the destination for new elements.
- Solution Context (/solution-context/) indicates the how the solution interacts with the customer's context.
- Solution Backlog (/program-and-solution-backlogs/) contributes direction and guidance to the vision.
- Continuous evolution of the Architectural Runway (/architectural-runway/) supports current and near-term Features (/features-and-capabilities/).
- Strategic Themes (/strategic-themes/) provide direction and serve as a decision-making filter
- Finally, and not to forget the obvious, the foremost experts in the domain are typically the Agile teams themselves. Primarily through the Product Owner (/product-owner/), teams continuously communicate emerging requirements and opportunities back into the program vision.

Capturing Vision in Solution Intent

Given the SAFe practice of cadence-based, face-to-face PI planning, vision documentation (various forms of which can be found in References 2, 3, and 4) is augmented, and sometimes replaced, by rolling-wave vision briefings. These provide routine, periodic presentations of the short- and longer-term vision to the teams. During PI planning, large solution level stakeholders, such as Solution Management (/product-and-solution-management/), describe the current overall Solution vision, while Product Management (/product-and-solution-management/) provides the specific ART context and vision.

The relevant elements of the vision, along with details of the current and specific behaviors of the system, are captured in solution intent.

Program Vision

When using 'Full SAFe' or 'Large Solution SAFe', each ART will likely have its own vision, detailing the direction of the specific capabilities or subsystems that it produces. This vision should be tightly coupled to the solution vision it supports.

Roadmap View

Having a sense of direction is critical to planning and engagement. But unless there is some realistic plan for how teams intend to fulfill the vision, people won't actually know what they must do. That purpose is filled by the Roadmap (/roadmap/). Figure 3 provides an example.

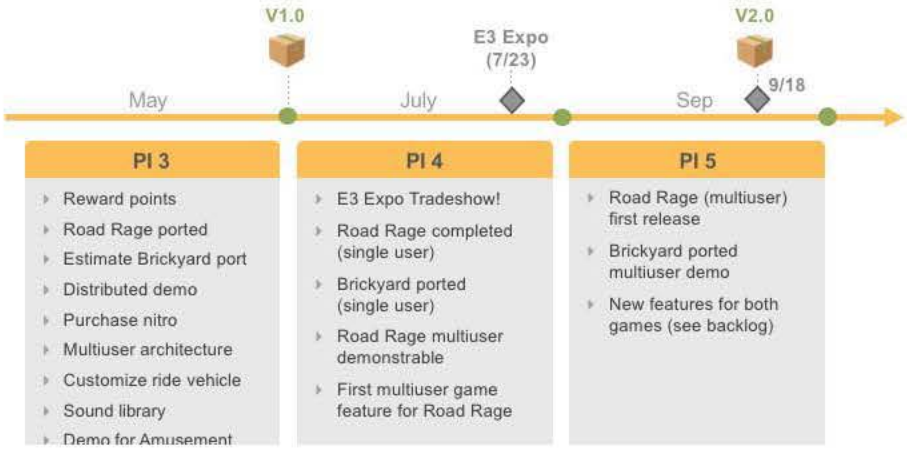


Figure 3. The roadmap is part of the vision

PI Planning Vision—the Top Ten Features

The roadmap is indeed helpful. But for action and execution, the immediate steps must be clear. Product and Solution Management have the responsibility to provide the direction for these next steps. In the SAFe context, this translates to a series of Increment steps forward, one PI (/program-increment/) at a time, and one feature at a time, as Figure 4 illustrates.

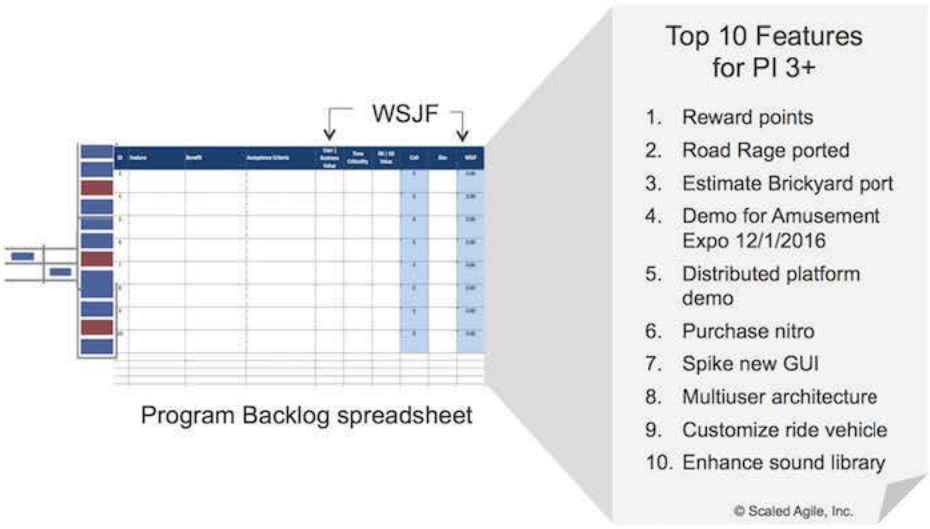


Figure 4. Vision is achieved one PI at a time, via the "Top 10 features for the next PI"

To achieve this, Product Management constantly updates feature priorities using WSJF (/wsjf/). Then, during PI planning, they present the "Top 10" to the team. The team won't be surprised by the new list, as they have seen the vision evolve over time and are aware of the new features that are headed their way. Further, the Program Kanban (/program-and-solution-kanbans/) is used to explore the scope of features, its benefit hypothesis, and acceptance criteria, so when features reach this boundary, they are fairly well formed and vetted. Architecture/Engineering (/system-and-solution-architect-engineering/) has already reviewed them, and various Enablers (/enablers/) have already been implemented.

However, everyone understands that the Top 10 is an *input*, not an *output* to the planning process, and what can be achieved in the next PI is subject to capacity limitations, dependencies, knowledge that emerges during planning, and more. Only the teams can plan and commit to a course of action, one that is summarized at every Program Increment (/program-increment/) in the PI Objectives (/pi-objectives/).

But these features are ready for implementation. And feature by feature, the program marches forward toward the accomplishment of the vision.

Solution Management presents a similar "Top 10" Capabilities (/features-and-capabilities/) list during Pre-PI Planning to align the ARTs within a Solution Train.

Learn More

[1] Heath, Chip and Dan Heath. *Switch: How to Change Things When Change Is Hard*. Broadway Books, 2010.

[2] Leffingwell, Dean. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison-Wesley, 2011.

[3] Leffingwell, Dean. *Scaling Software Agility: Best Practices for Large Enterprises*. Addison-Wesley, 2007.

[4] Leffingwell, Dean and Don Widrig. *Managing Software Requirements*. Addison-Wesley, 2001.

Last update: 17 June, 2017

FRAMEWORK

Downloads (/posters/)

Latest Updates (/blog/)

TRAINING

Course Calendar (http://www.scaledagileacademy.com/events/event_list.asp)

About Certification (http://www.scaledagileacademy.com/?page=WhichCertification)

Become a Trainer (http://www.scaledagileacademy.com/?page=becomeatrainer)

PERMISSIONS

Permission FAQ (http://www.scaledagile.com/permission-faq/)

Permissions Form (http://www.scaledagile.com/permissions-form/)

Usage and Permissions (/usage-and-permissions/)

PARTNER

Becoming a Partner (http://www.scaledagile.com/become-a-partner/)

Partner Directory (http://www.scaledagile.com/listingcategory/directory/)

Partner Event Calendar (http://www.scaledagile.com/event-list/)

GET SOCIAL

Twitter (https://twitter.com/ScaledAgile)

Linkedin (https://www.linkedin.com/grps/Scaled-Agile-Framework-4189072?)

YouTube (https://www.youtube.com/user/scaledagile)

SlideShare (http://www.slideshare.net/ScaledAgile)

RECENT POSTS

Updated SAFe Scrum Master and SAFe Advanced Scrum Master courses now available! (/blog/updated-safe-scrum-master-and-safe-advanced-scrum-master-courses-now-available/)

Sep, 19th 2017

SAFe from the Trenches at Agile Australia (/blog/safe-from-the-trenches-at-agile-australia/)

Sep, 19th 2017

Updated SAFe Release Train Engineer and SAFe Product Owner/Product Manager courses now available! (/blog/updated-safe-release-train-engineer-and-safe-product-ownerproduct-manager-courses-now-available/)

Aug, 31th 2017

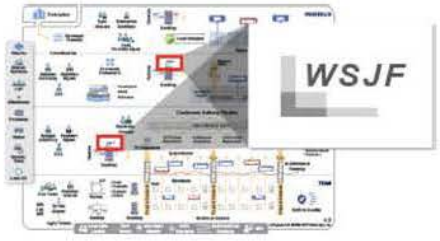
SCALED AGILE, INC

CONTACT US

5480 Valmont Rd., Suite 100

Boulder, CO 80301 USA

Email: support@scaledagile.com (mailto:support@scaledagile.com)



(/)

> *If you only quantify one thing, quantify the cost of delay.*
>
> —Don Reinertsen

WSJF (Weighted Shortest Job First)

Weighted Shortest Job First (WSJF) is a prioritization model used to sequence "jobs" (e.g., Features, Capabilities, and Epics) to produce maximum economic benefit. In SAFe, WSJF is estimated as the cost of delay divided by job size.

Agile Release Trains (/agile-release-train/) (ARTs) provide an ongoing, continuous *flow* of work that makes up the enterprise's incremental development effort. It avoids the overhead and delays caused by the start-stop-start nature of traditional projects, where authorizations and phase gates control the program and its economics.

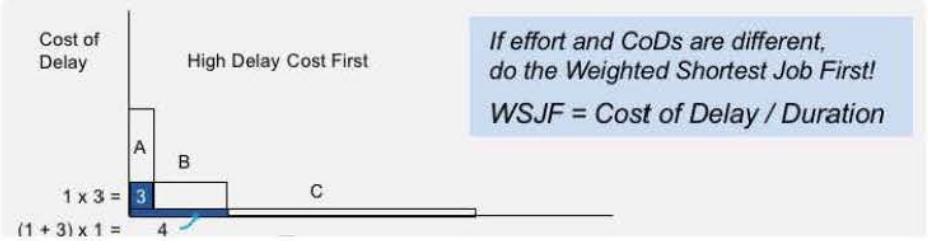
While this continuous flow model speeds the delivery of value and keeps the system Lean, priorities must be updated continuously to provide the best economic outcomes. In flow, job *sequencing*, rather than theoretical, individual job ROI, produces the best result. To that end, Weighted Short Job First (*WSJF*) is used to prioritize backlogs by calculating the relative *cost of delay* and *job size* (a proxy for duration). Using WSJF at Program Increment (/program-increment/) boundaries continuously updates backlog priorities based on user and business value, time factors, risk, opportunity enablement, and effort. WSJF also conveniently and automatically *ignores sunk costs*, a key principle of Lean economics.

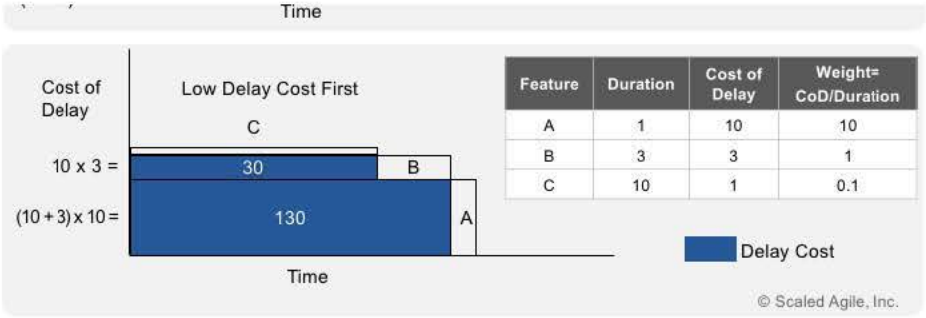
Details

Reinertsen [2] describes a comprehensive model called "Weighted Shortest Job First" (*WSJF*) for prioritizing jobs based on the economics of product development flow. WSJF is calculated as the *cost of delay divided by job duration*. Jobs that can deliver the most value (or cost of delay) and are of the shortest duration are selected first for implementation. When applied in SAFe, the model supports a number of additional key principles of product development flow, including:

- Take an economic view
- Ignore sunk costs
- If you only quantify one thing, quantify the cost of delay
- Economic choices must be made continuously
- Use decision rules to decentralize economic control

The impact of properly applying WSJF can be seen in Figure 1. (See [2] for full discussion.) The areas shaded in blue illustrate the total cost of delay in each case. Doing the *weighted shortest job first* delivers the best economics.





From *The Principles of Product Development Flow*, by Donald G. Reinertsen, Celeritas Publishing, © 2009 Donald G. Reinertsen

Figure 1. The economic effect of doing the Weighted Shortest Job First (WSJF): cost of delay for work

Calculating the Cost of Delay

In SAFe, our "jobs" are the Epics (/epic/) and the Features and Capabilities (/features-and-capabilities/) we develop, so we need to establish both the cost of delay and the duration for each job. There are three primary elements that contribute to the cost of delay:

User-Business Value: Do our users prefer this over that? What is the revenue impact on our business? Is there a potential penalty or other negative impact if we delay?

Time Criticality: How does the user/business value decay over time? Is there a fixed deadline? Will they wait for us or move to another solution? Are there Milestones (/milestones/) in the critical path impacted by this?

Risk Reduction-Opportunity Enablement Value: What else does this do for our business? Does it reduce the risk of this or a future delivery? Is there value in the information we will receive? Will this feature open up new business opportunities?

Moreover, since we are in continuous flow and should have a large enough backlog to choose from, we needn't worry about the absolute numbers. We can just compare backlog items relative to each other using the modified Fibonacci numbers we use in estimating poker. Then the relative cost of delay for a job is:

Cost of Delay = User-Business Value + Time Criticality + Risk Reduction and/or Oppty Enablement

Duration

Next we need to understand job duration. That can be pretty difficult to figure, especially since early on we perhaps don't yet know who is going to do the work or what capacity allocation they might be able to give it. So we probably don't really know. Fortunately, we have a ready proxy: job *size*. In systems with fixed resources, job size is a good proxy for duration. (If I'm the only one mowing my lawn, and the front yard is three times bigger than the back yard, the front lawn is going to take three times longer to mow.) And we know how to estimate item size in story points already (see Features (/features-and-capabilities/)). Taking job size, we have a reasonably straightforward calculation for comparing jobs via WSJF, as Figure 2 illustrates:

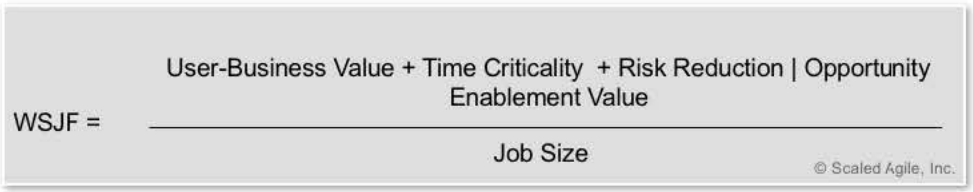


Figure 2. A formula for calculating WSJF

Then, for example, we can create a simple table to compare jobs (three jobs in this case), as shown in Figure 3:

Feature	User-Business Value	Time Criticality	RR-OE Value	Job Size	WSJF

- Rate each parameter of each feature against the other features.
- Scale: 1, 2, 3, 5, 8,13, 20.
- Do one column at a time, start by picking the smallest item and giving it a "1". There must be at least one "1" in each column!
- The highest priority is the highest WSJF.

Figure 3. A sample spreadsheet for calculating WSJF

To use the table, the team rates each job relative to other jobs on each of the three parameters. (Note: With relative estimating, you look at one *column* at a time, set the smallest item to a one, and then set the others relative to that item.) Then divide by the size of the job (which can be either a relative estimate or an absolute number based on the estimates contained in the backlog) and calculate a number that ranks the job's priority.

The job with the highest WSJF is the next most important item to do.

One outcome of this model is that really big, important jobs have to be divided into smaller, pretty important jobs in order to make the cut against easier ways of making money (i.e., small, low-risk jobs that your Customers (/customer/) are willing to pay for now). But that's just Agile at work. Since the implementation is incremental, whenever a continuing job doesn't rank well against its peers, then you have likely satisfied that particular requirement sufficiently that you can move on to the next job.

As we have described, another advantage of the model is that it is *not* necessary to determine the absolute value of any of these numbers. Rather, you only need to rate the parameters of each job against the other jobs from the same backlog.

Finally, as the backlog estimates should include only the job size remaining, then frequent reprioritization means that the system will *automatically ignore sunk costs*.

A Note on Job Size as a Proxy for Duration

However, we do have to be careful about the proxy we chose for duration. If availability of resources means that a larger job may be delivered more quickly than some other item with about equal value, then we probably know enough about the job to use *duration* to have a more accurate result. (If I can get three people to mow my front lawn while I mow the back, then these items have about the same duration, but not the same cost.) But this is rarely necessary in the flow of value, in part because if there is some small error in selection, that next important job will make its way up soon enough.

Learn More

[1] Leffingwell, Dean. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison-Wesley, 2011.

[2] Reinertsen, Don. *Principles of Product Development Flow: Second Generation Lean Product Development*. Celeritas Publishing, 2009.

Last update: 17 June, 2017

PERMISSIONS

Permission FAQ (http://www.scaledagile.com/permission-faq/)

Permissions Form (http://www.scaledagile.com/permissions-form/)

Usage and Permissions (/usage-and-permissions/)

PARTNER

Becoming a Partner (http://www.scaledagile.com/become-a-partner/)

Partner Directory (http://www.scaledagile.com/listingcategory/directory/)

Partner Event Calendar (http://www.scaledagile.com/event-list/)

GET SOCIAL

Twitter (https://twitter.com/ScaledAgile)

Linkedin (https://www.linkedin.com/grps/Scaled-Agile-Framework-4189072?)

YouTube (https://www.youtube.com/user/scaledagile)

SlideShare (http://www.slideshare.net/ScaledAgile)

RECENT POSTS

Updated SAFe Scrum Master and SAFe Advanced Scrum Master courses now available! (/blog/updated-safe-scrum-master-and-safe-advanced-scrum-master-courses-now-available/)

Sep, 19th 2017

SAFe from the Trenches at Agile Australia (/blog/safe-from-the-trenches-at-agile-australia/)

Sep, 19th 2017

Updated SAFe Release Train Engineer and SAFe Product Owner/Product Manager courses now available! (/blog/updated-safe-release-train-engineer-and-safe-product-ownerproduct-manager-courses-now-available/)

Aug, 31th 2017

SCALED AGILE, INC

CONTACT US

5480 Valmont Rd., Suite 100
Boulder, CO 80301 USA

Email: support@scaledagile.com (mailto:support@scaledagile.com)

Phone: +1.303.554-4367

BUSINESS HOURS

Weekdays: 9am to 5pm
Weekends: CLOSED