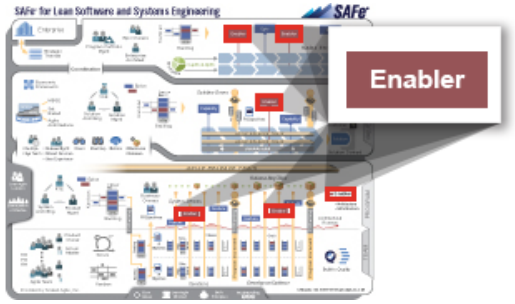

(/)

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

Enablers Abstract

Enablers are technical initiatives meant to enable and support the development of business initiatives. Enablers (reflected in red on the Big Picture) exist on all four levels of SAFe: *Enabler Epics* at the Portfolio Level, *Enabler Capabilities* at the Value Stream Level, *Enabler Features* at the Program Level, and *Enabler Stories* at the Team Level. Enablers can be used for any activities that are necessary to support upcoming business features, but generally they fall into one of three categories:

1. Exploration – Exploration enablers are used to build understanding of what is needed by the Customer, to understand prospective Solutions, and to evaluate alternatives
2. Architecture – Architectural enablers are used to build the Architectural Runway in order to enable smoother and faster development
3. Infrastructure – Infrastructure enablers are used to build and enhance the development and testing (and occasionally deployment) environments, thereby facilitating faster development and higher-quality testing

Details

Enablers are work items that capture and bring visibility to all the work necessary to support efficient development and delivery of future business Features (/features-and-capabilities/). They are used primarily for exploration, system and Solution (/solution/) architectural evolution, and to enhance development and testing environments (see later sections of this article). Since they reflect real work, and sometimes plenty of it, they cannot be invisible. They are treated like all other value-added development activities and are thereby subject to estimating, visibility and tracking, WIP limits, feedback, and, finally, presentation of results.

Types of Enablers

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

- ***Enabler epics*** are a type of Epic (/epic/), and as such are written using the value statement format defined for epics. They tend to cut across Value Streams (/value-streams/) and Program Increments (PIs) (/program-increment/). They must include a lightweight business case to support their implementation and are identified and tracked through the Portfolio Kanban (/portfolio-kanban/) system.
- ***Enabler capabilities and features*** occur at the Value Stream (/value-stream-level/) and Program Levels (/program-level/), where they capture work of that type. As these enablers are a type of Feature or Capability (/features-and-capabilities/), they share the same attributes, including a statement of benefits and acceptance criteria, and they must be structured so as to fit within a single PI.
- ***Enabler stories***, as a type of Story (/story/), must fit in Iterations (/iterations/). However, while they may not require user voice format, they have acceptance criteria to clarify the requirements and support testing.

Enablers are written, prioritized and follow the same rules as their respective epics, capabilities, features, and stories. For example: Features and enabler features are written using a Features and Benefits matrix, have acceptance criteria, use story points for estimation and WSJF (/wsjf/) for prioritization, etc.

Creating and Managing Enablers

Most enablers are created when business initiatives, making their way through the different Kanban systems, require exploration enablers to validate the need or solution; architectural enablers to pave the runway; or infrastructure enablers to be ready to develop, test, and integrate the initiatives.

Enablers are often created by architects or by systems engineering at the various levels, whether by Enterprise Architects (/enterprise-architect/) at the Portfolio Level (/portfolio-level/) or by Solution and System Architects/Engineering (/system-and-solution-architect-engineering/) at the value stream and program levels. The architects who create the enablers steer them through the Kanban systems, providing both the guidance needed to analyze them and the information needed to estimate and implement them.

Some enablers will emerge locally from the needs of the Agile Teams (/agile-teams/), Agile Release Trains (ARTs) (/agile-release-train/), or

Some enablers will emerge locally from the needs of the Agile Teams (/agile-teams/), Agile Release Trains (ARTs) (/agile-release-train/), or value streams to improve the existing solution. Enablers that make it through the Kanban systems will be subject to capacity allocation in the Program and Value Stream Backlogs (/program-and-value-stream-backlogs/) to ensure that enough emphasis is placed on both, furthering the solution and extending the Architectural Runway (/architectural-runway/). Capacity allocation can be applied for enabler work as a whole or it can differentiate between the various 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, since at the beginning of development little is known about exactly what the Customer (/customer/) needs or how to implement it. Often the customers themselves don't understand exactly what they want, and only through iterative product development and demos do they gain an understanding of what they actually need.

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

Architecture

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

Some of these 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. They 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 PI for the Solution Demo (/solution-demo/). Many Enterprises (/enterprise/) even implement Continuous Integration (/continuous-integration/) to ensure that the solution is always running and to reduce the risk at the integration points.

To support these frequent or continuous integration and testing constructs, there is a need to develop infrastructure at the Team Level (/team-level/), program level, and value stream level. New infrastructure is also sometimes needed to increase the rate of Release (/release/) to the Customer. Agile Teams, working with the System Team (/system-team/), are responsible for building and maintaining this infrastructure. 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.

Implement Architectural Enablers Incrementally

Architectural enabler work can be big. It is important to remember that it needs to be broken down into small enabler stories that can fit in iterations. This can be difficult, 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 such 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 [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; do no harm.

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

Enablers drive better economics by creating the technology platforms that deliver business functionality. But innovative product development cannot occur without risk-taking. Therefore, initial technology-related decisions cannot always be correct. Hence the Agile enterprise must be prepared to reverse course on occasion. The Ignore Sunk Costs principle of product development flow ([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 respectively. During the analysis phase of the Kanban system, one of the important decisions to make is whether to implement the enabler in all VSs/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 caused by not having the full enabler, as Figure 2 illustrates.

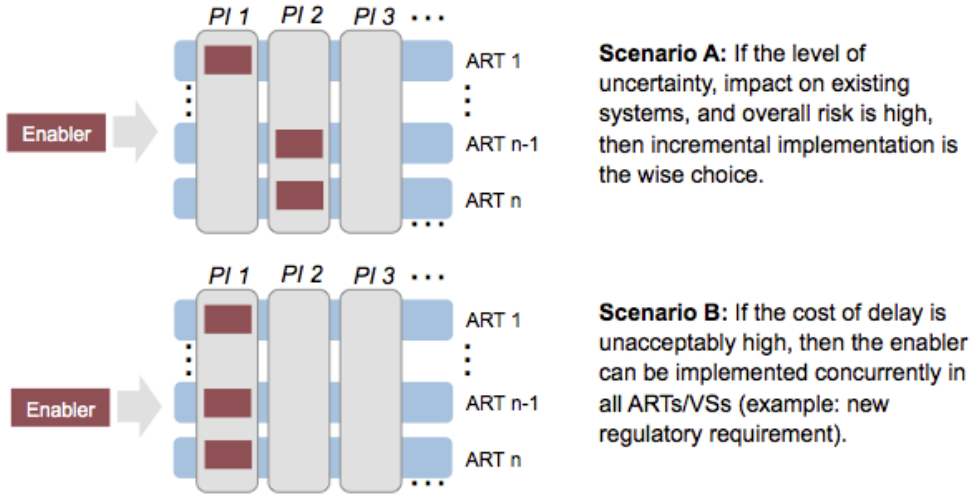

Figure 2. Two scenarios for implementing large enablers spanning multiple ARTs or VSs

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: 21 April 2016

FRAMEWORK

Downloads (/posters/)

Latest Updates (/blog/)

SAFe 3.0 (http://v3.scaledagileframework.com)

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

New Essential SAFe guidance article (/new-essential-safe-guidance-article/)

Nov, 30th 2016

SAFe Website Updates and new Glosssary (/safe-website-updates-and-new-glosssary/)

Nov, 29th 2016

End of life for SAFe 3.0 website and courseware 12/30/16 (/eol-safe3/)

Nov, 28th 2016

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