Scaling Agile – The SAFe Framework

Kim Horn

Version 0.5 **10 May 2018**

https://www.scaledagileframework.com
https://www.scaledagile.com

Some questions about our 'process' ?

1. Is it understood by all, is it approachable, well documented ?
2. Is it complete and consistent; is the 'lifecycle' clearly defined ?
   - Does it integrate: development, dev-ops, architecture, program, portfolios, HR, QA, strategy ?
   - Does it include all the roles required; do people really know what they have to do, when ?
3. Does it scales from small to large, and upward:
   - At all levels of the organization, and across team structures ?
   - Is quality managed at all levels ?
4. Is it configurable to match different goals ?
5. Do we understand how Governance happens, does it work, is autonomy working ?
6. Do we produce useful requirements, are they well understood, do they at work at multiple levels ?
7. Do we utlise the best from Agile and Lean ?
8. Are all the dependencies understood to deliver on time, without delays?
9. Do we have the capability to design and manage the process, the collateral and change:
   - is it our core business, do we do this well ?
10. **Maturity - Do we want to try and improve productivity further, have better motivated people, less defects, and faster time to market ? Do we have the right measures and process in place to do this improvement ?**
11. **Do we want our business as a whole to be more 'agile', at all levels ?**

Why Use SAFE – 11 Reasons ?

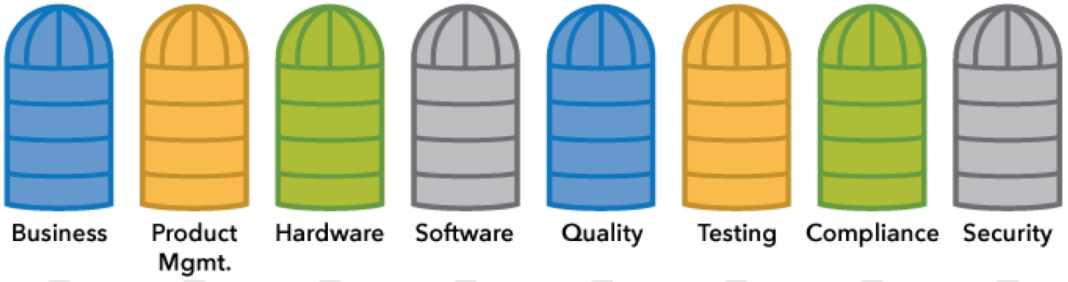
The Scaled Agile Framework, or SAFe® for short, is a "publicly available framework for applying lean/agile practices at Enterprise scale."

1. It is publicly available and free to use.
2. It is available in a highly approachable, usable form.
3. It is lightweight, in the best sense of the term.
4. It is practical.
5. It is specific.
6. It conveniently codifies some of the most common agile practices in use today.
7. It offers useful extensions to common agile practices.
8. It grounds agile practices in an enterprise context.
9. It offers a complete picture of software development.
10. It is regularly maintained.
11. It is supported by 250+ Scaled Agile Partners for training, consulting, implementation services.

Core concept - Agile Release Train (ART)

SAFe outlines a consistent approach to planning, execution and delivery of value, called an ART.

An ART is a lightweight “program container” that brings multiple Agile teams together on a consistent cadence every 8-12 weeks known as a **Program Increment (PI).** ARTS are cross-functional and have all the people they need to deliver value.



At the beginning of each PI, the ART comes together to plan what they will deliver in that PI. This opportunity to work together as a team of teams helps organizations uncover, plan for, and address cross-team dependencies, risks, and impediments.

During the PI, the teams on the ART use well-known practices like the Scrum-of-Scrums to remain in sync on cross-team dependencies and new practices like cross-team System Demos every two weeks to inspect and adapt the product from a customer-centric point of view.

4 Types of SAFe Configurations - Introduce Competencies

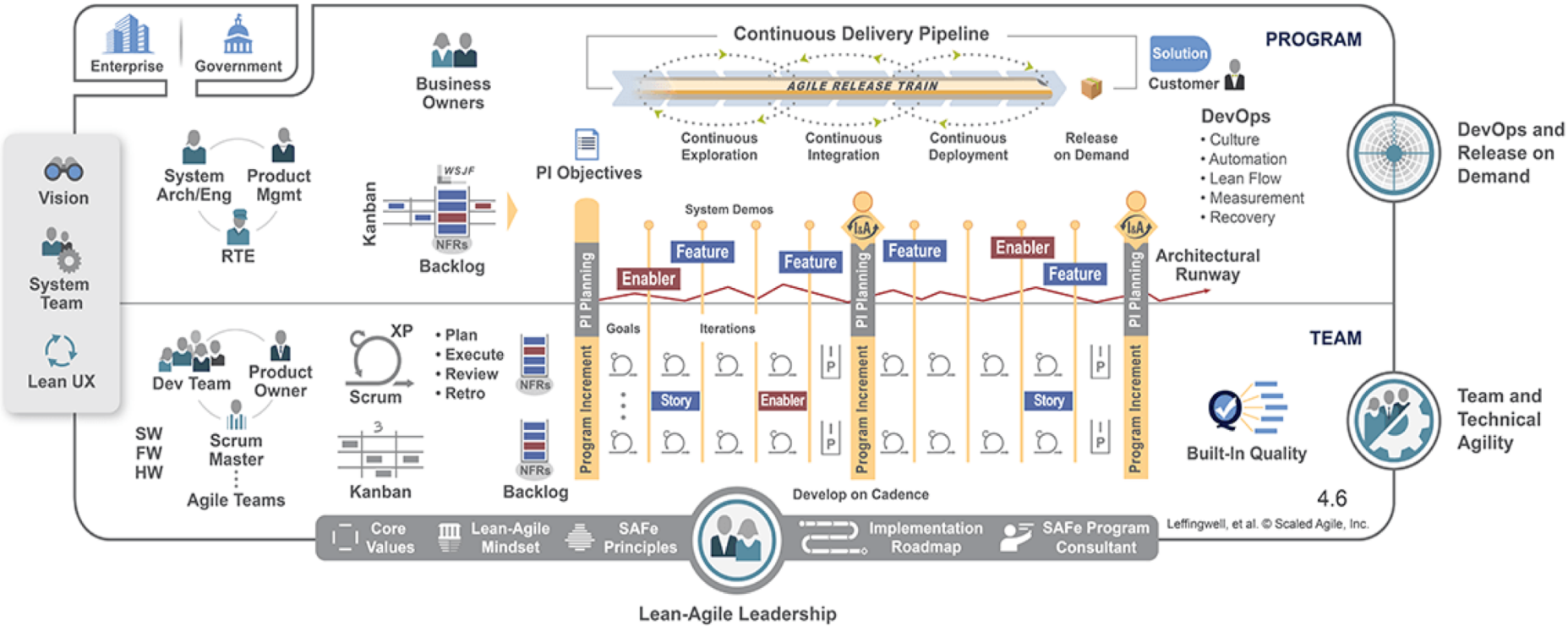
1. **Essential SAFe** - the most basic configuration. It provides a starting point for implementing SAFe and describes the most critical elements needed to realise the majority of the framework's benefits. Provides the Team and Technical Agility, DevOps and Release on Demand, and Lean-Agile Leadership competencies.

2. **Portfolio SAFe** - provides the Lean Portfolio Management competency which aligns portfolio execution to enterprise strategy. It organizes development around the flow of value through one or more value streams

3. **Large Solution SAFe** - introduces the Business Solutions and Lean Systems Engineering competency, which supports those building the largest and most complex solutions that require multiple Agile Release Trains and Suppliers, but do not require portfolio-level considerations.

4. **Full SAFe** - includes all Five Core Competencies of the Lean Enterprise. It is the most comprehensive version of the Framework and supports enterprises that build and maintain a portfolio of large and complex solutions.

Essential Configuration – 2 levels only



The Two Levels

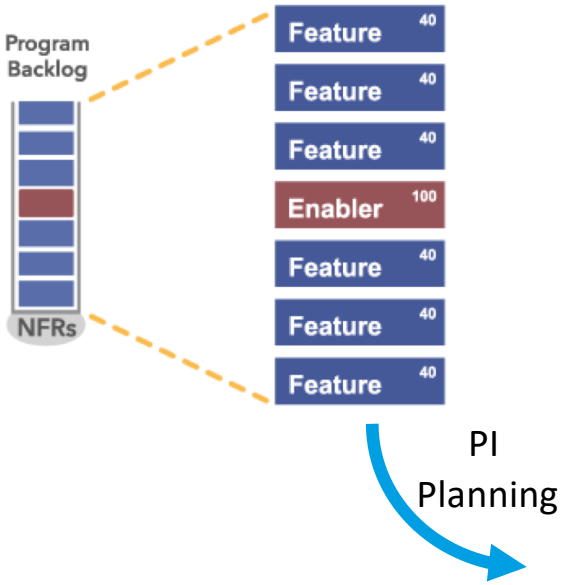
Program:

- Contains the roles and activities needed to continuously deliver solutions via an **Agile Release Train (ART).**
- The ART is a long-lived team of Agile teams, which, along with other stakeholders, develops and delivers solutions incrementally, using a series of fixed-length Iterations within a **Program Increment (PI)** timebox.
- The ART aligns Agile Teams to a common business and technology mission.
- **Continuous Delivery Pipeline** – the workflows, activities, and automation needed to provide a constant release of value to the end user. Each ART has a known velocity.
- **DevOps** – A mindset, culture, and a set of technical practices. It provides communication, integration, automation, and close cooperation among all the people needed to plan, develop, test, deploy, release, and maintain a solution.

Team:

- Contains all the roles, activities, events, and processes which Agile Teams build and deliver value;
- Responsible for defining, building, and testing Stories from their Team Backlog;
- May be multiple Agile Teams within a program;
- Apply Scrum or XP or Kanban.

2 Backlogs: Program to Team – A Consistent Planning Mechanism

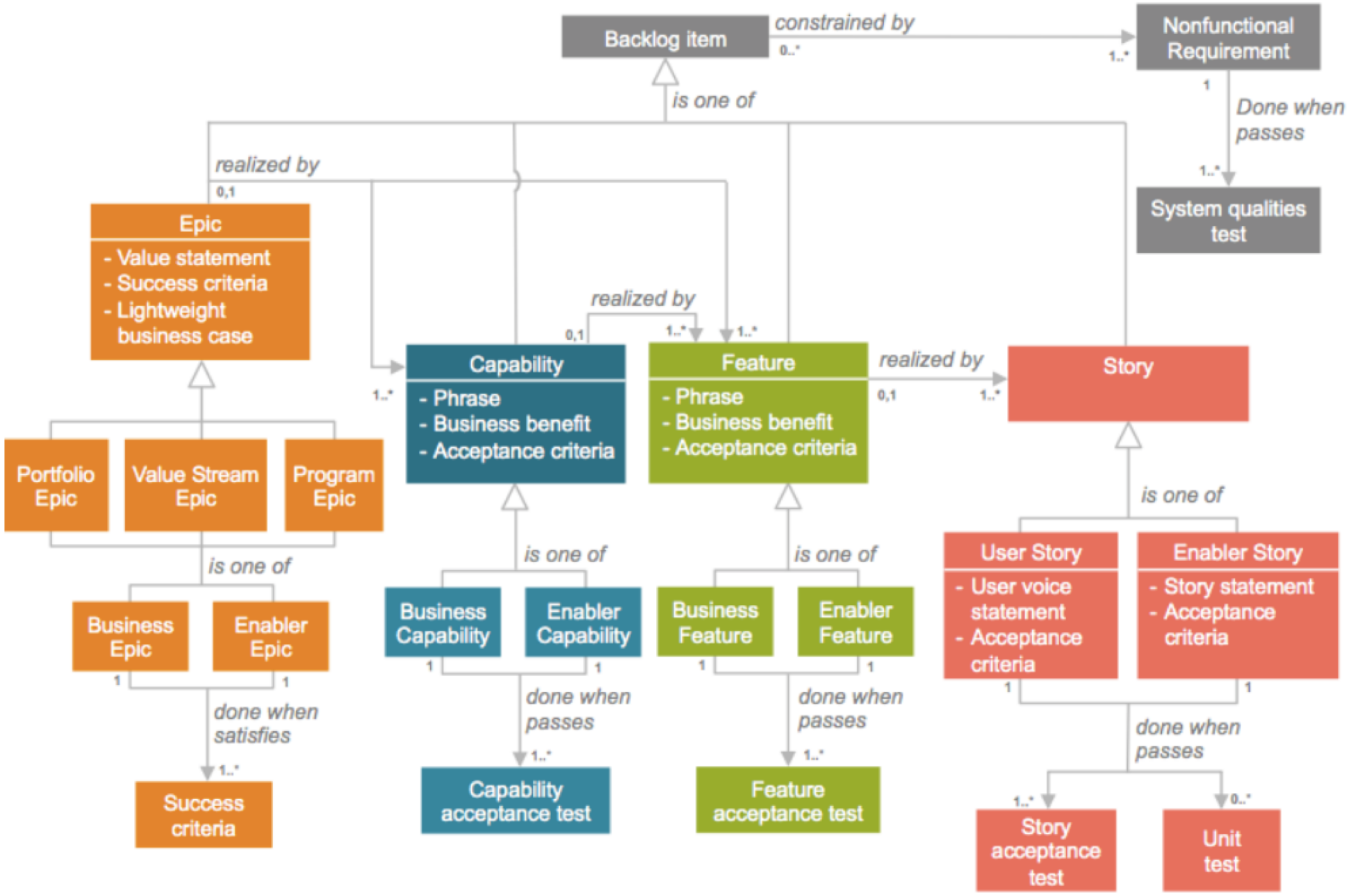


The 2 Backlogs (Program and Team) include:

- Enablers
- Program Epics & Features
- User Stories
- Refactoring Items
- Maintenance Items
- NFRs - Non-functional requirements constrain the Backlog, and occur at all levels. They need to be eventually tested.

Capacity Allocation determines the balance of work on which backlog items. For example how much Enabler vs Feature work

Requirements and Backlog Model

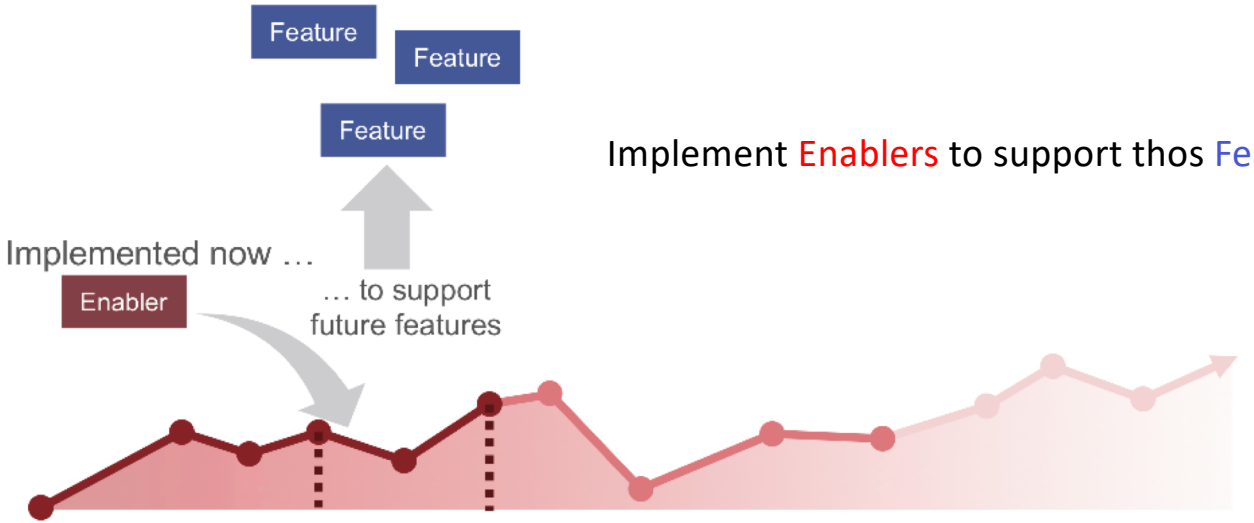


Unites: Non-Functional, Traditional and Agile Requirements, at various levels of granularity and organization levels, into backlog model for execution and planning.

Architecture Runway enables Fast Iterative Feature Delivery

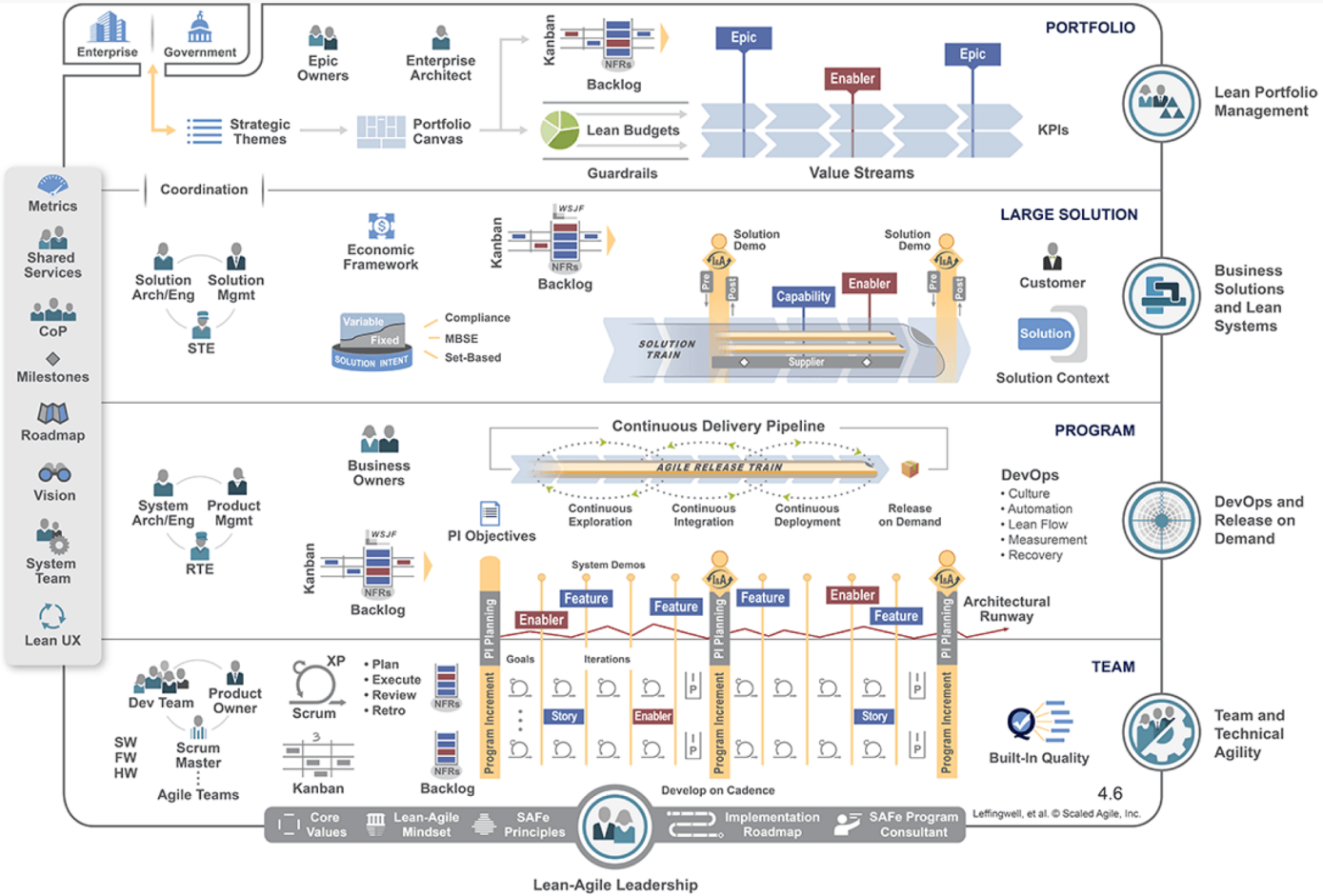
Architecture Runway provides 'just enough' technical enablement to keep velocity high and avoid excessive redesign and delays. Bigger systems often require more runway.

Agile Architecture builds out the runway**;** it is a set of values and practices that support the active evolution of the design and architecture of a system while implementing new system capabilities.



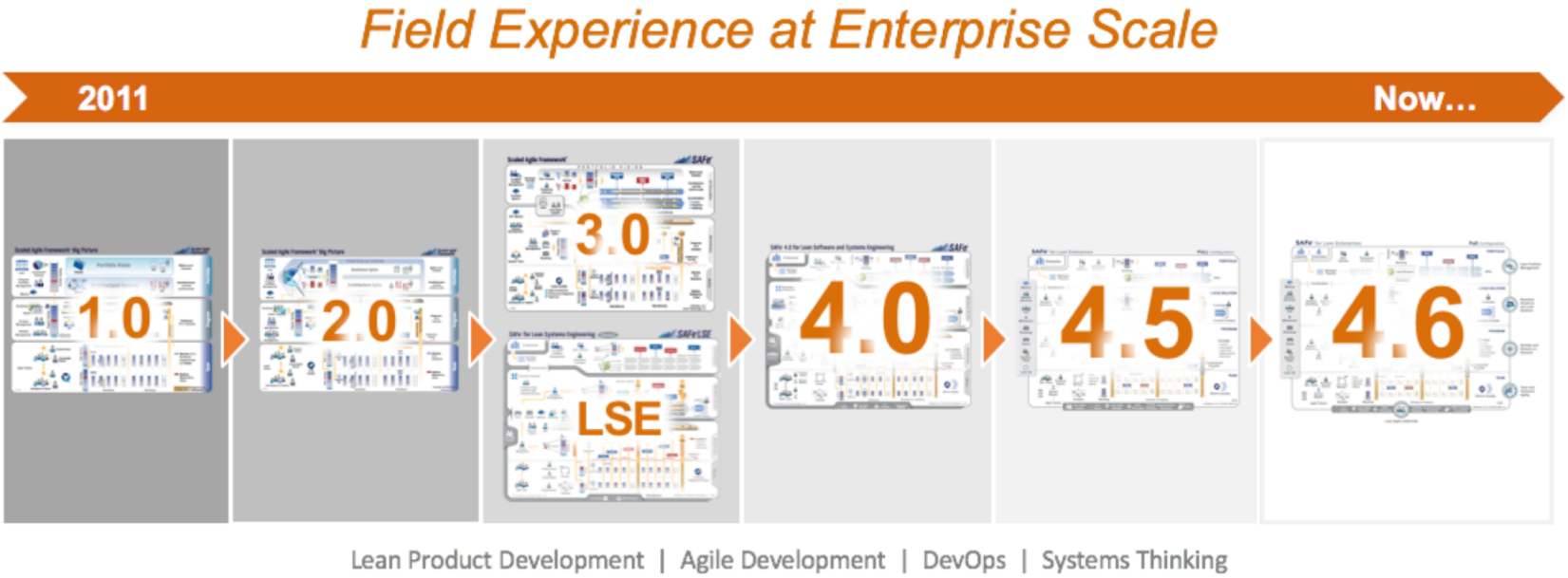
Implement Enablers to support thos Features coming up.

The Full Model

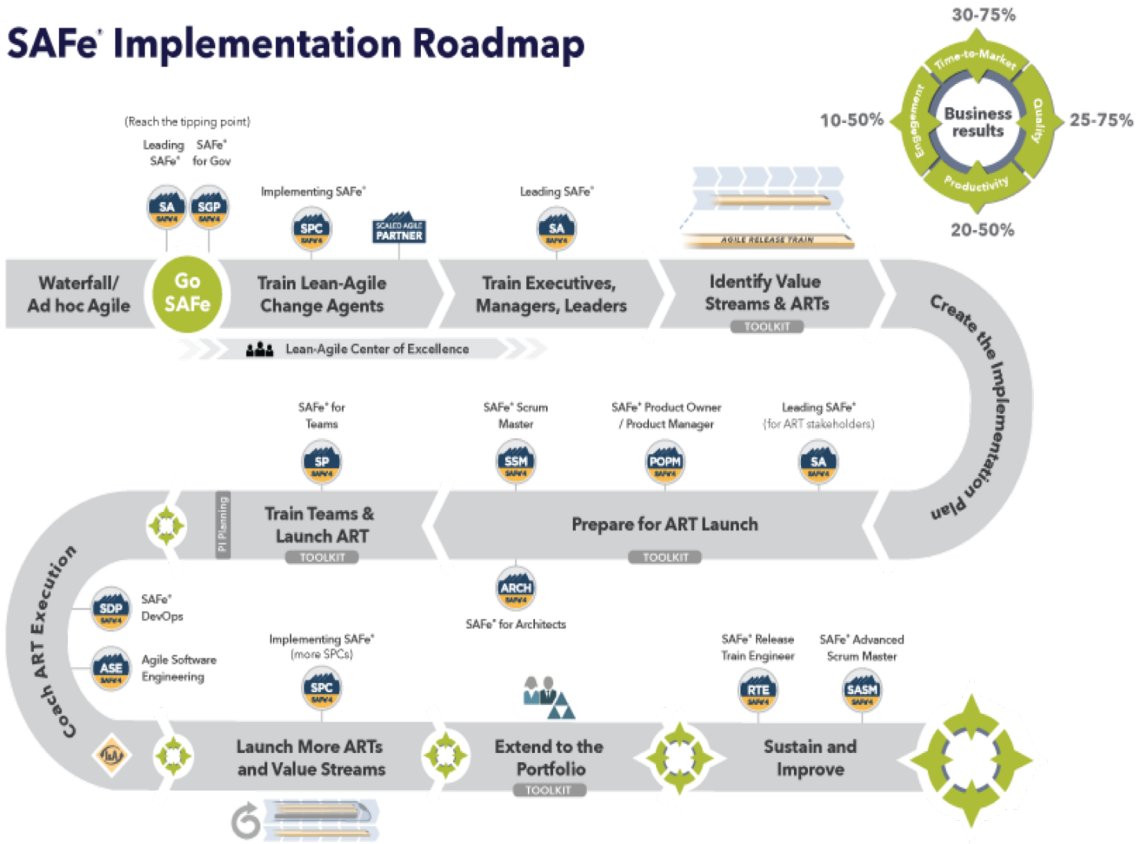


SAFe is Continually Improving

SAFe has grown with the marketplace as we uncover new and better ways of developing software and systems



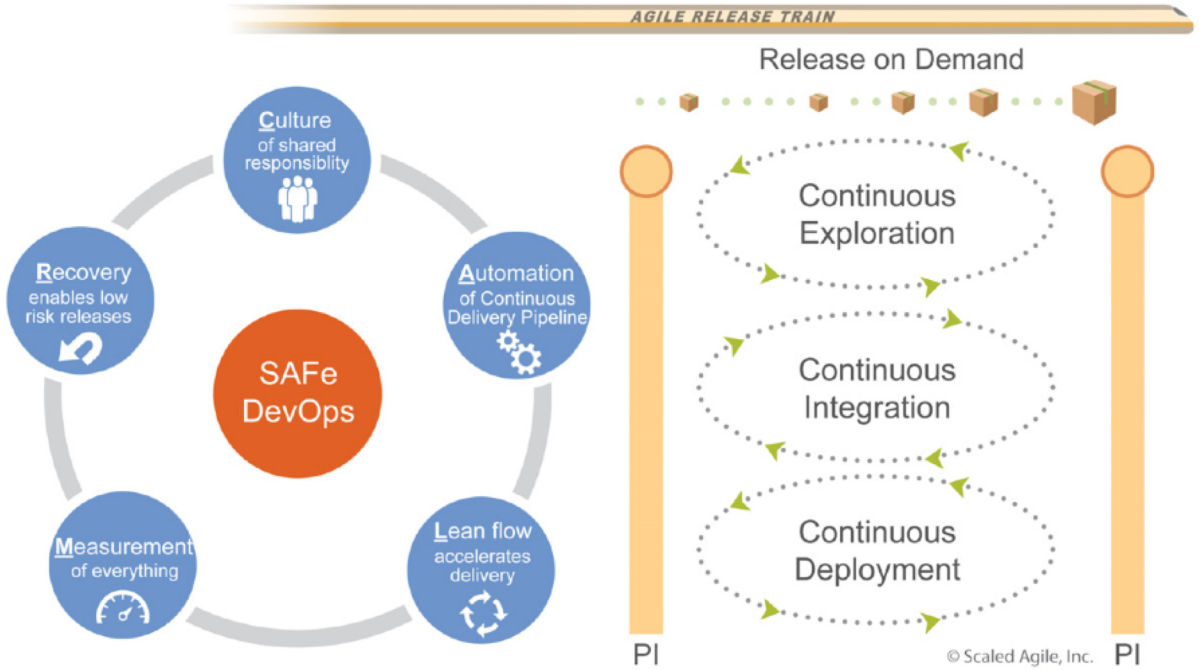
Implementation Roadmap



The End

ARTs apply DevOps (CI/CD); Releasability

DevOps improves collaboration and flow between Development and IT Operations with a continuous delivery pipeline.



The System Team is a specialized Agile Team that assists in building and using the Agile development environment, including Continuous Integration, test automation, and Continuous Deployment. The System Team supports the integration of assets from Agile teams, performs end-to-end Solution testing where necessary, and assists with deployment and release.

Architectural Runway – Implements Architectural Strategy

The **Architectural Runway** consists of the existing code, components, and technical infrastructure needed to implement near-term features without excessive redesign and delay.

- It provides the necessary technical foundation for developing business initiatives and implementing new Features and/or Capabilities.
- The development of new features and capabilities consumes the architectural runway, continual investment must be made to extend it by implementing **Enablers**.
- **Enablers** support the activities needed to extend the Architectural Runway to provide future business functionality. These include exploration, infrastructure, compliance, and architecture development. Enablers go on the backlogs.
- Some enablers fix existing problems with the Solution, such as improving the performance or User Experience

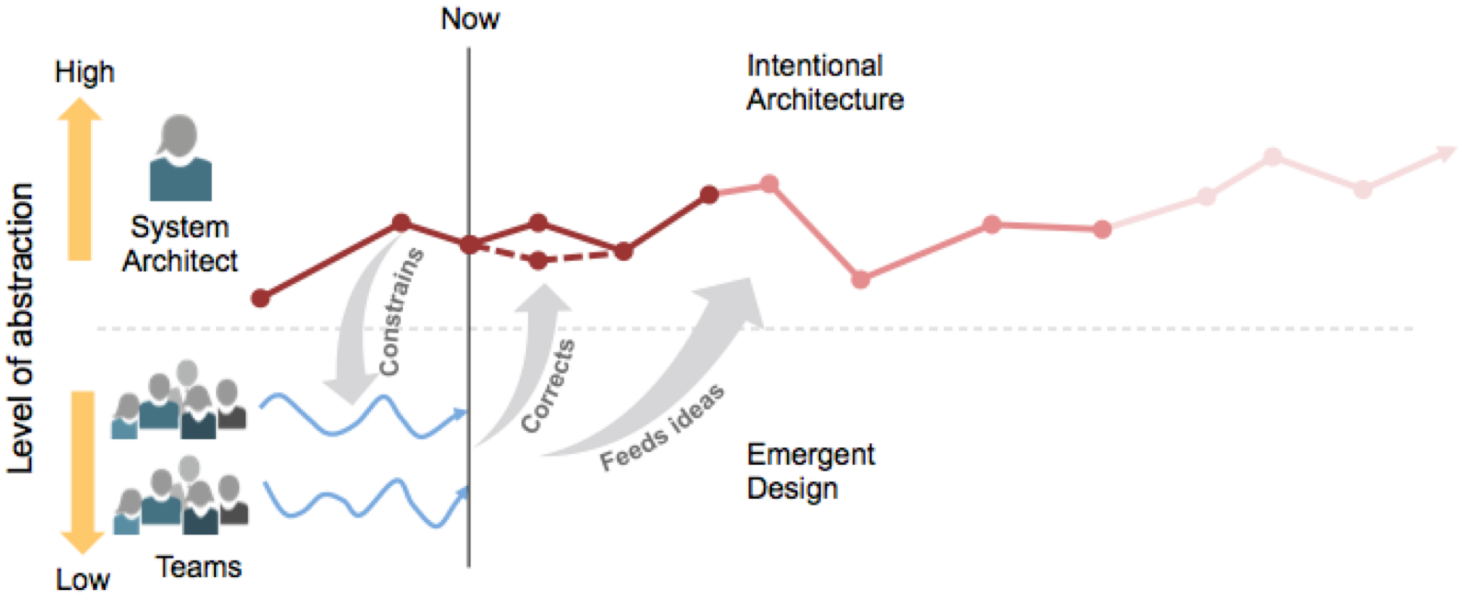
Without the Architecture Runway, **emergent design** fails to handle the complexity of large-scale system development, and the following problems start to occur:

- Excessive redesign and delays reduce velocity Systems become difficult to integrate, validate, and maintain;
- Decline of system qualities, known as Nonfunctional Requirements (NFRs);
- Reduced collaboration and synchronization among teams;
- Low reuse of common components and redundancy of solution elements.

Intentional Vs Emergent Architecture

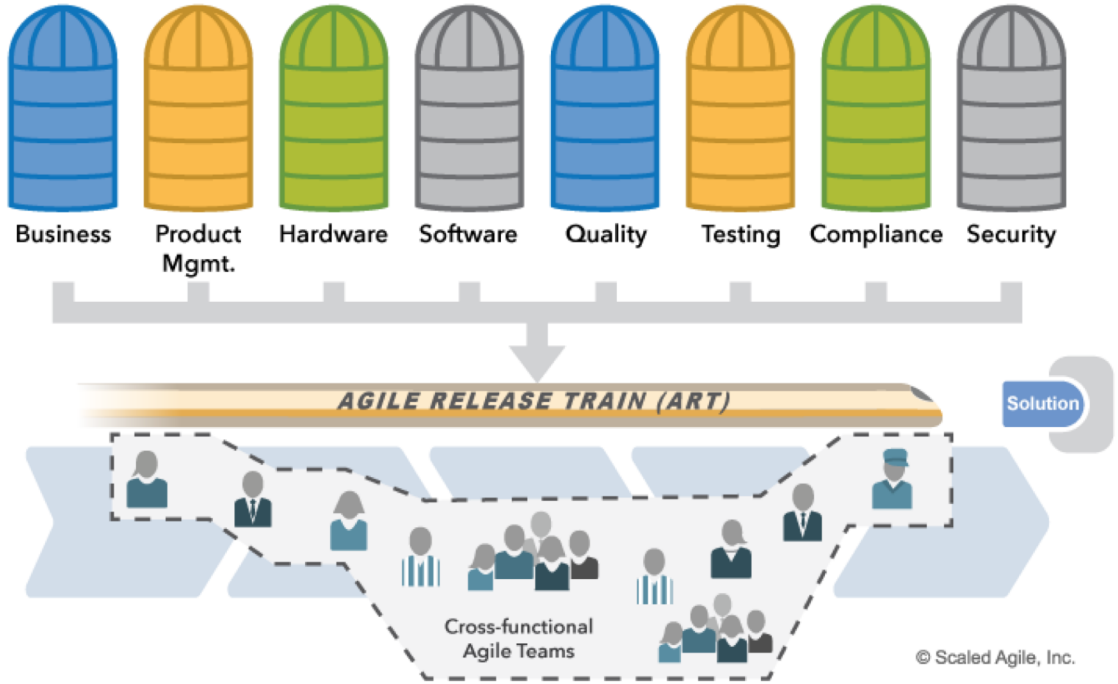
> "While we must acknowledge emergence in design and system development, a little planning can avoid much waste." —
>
> James Coplien, Lean Architecture

- Architecture can just emerge, Emergent Design, and it can be Intentional, with just enough up front work to provide direction.
- Intentional Architecture corrects emergent design via collaboration.



Agile Release Trains (ARTs) are Cross Functional

The ART applies systems thinking and builds a cross-functional organization that is optimized to facilitate the flow of value from ideation to deployment. ARTS have all the people they need to deliver value.



This is the inverse of traditional approach where management, organises the hand off of work, across functional silos. However, value does not flow easily and is hampered. Hand offs produce waste, and the silos are distanced from the business problem.

Program Events and Artefacts

The **Program** level uses three main activities to help coordinate the ART:

- **PI Planning** – A cadence-based, face-to-face planning event that serves as the heartbeat of the ART, aligning all the teams on the ART to the mission and vision.
- **System Demo** – Provides an integrated view of new features for the most recent iteration delivered by all the teams in the ART. Each demo provides ART stakeholders with an objective measure of progress during a PI.
- **Inspect & Adapt** – Is a significant event where the current state of the solution is demoed and evaluated. Teams then reflect and identify improvement backlog items via a structured problem-solving workshop.

Artefacts:

- **The Vision** is a description of the future state of the Solution under development. It reflects Customer and stakeholder needs, as well as the Feature and Capabilities, proposed to meet those needs.
- **Features and Epics;**
- **Program Backlog** – Holds the upcoming **Features**, and **Enablers** that build out the Architectural Runway.
- **Program Kanban** – It manages the flow of features and enablers through the Continuous Delivery Pipeline.
- **PI Objectives** – Specific business and technical goals that an ART intends to achieve in the next PI.
- **Architectural Runway** – Architecture work defined by **Enablers**.

Team Events

- **Iteration Planning** – Is an event in which an Agile team determines the Iteration Goals and how much of the team backlog they can commit to during an upcoming iteration. Team capacity determines the number of stories and enablers that are selected.
- **Iteration Review** – Is a cadence-based event in which the team inspects the increment at the end of the iteration and adjusts the team backlog based on feedback. All work done during the iteration is demoed during the iteration review.
- **Iteration Execution** – Is how the Agile team develops an increment of an effective, high-quality, working, tested system within the timebox. Agile team holds a 15-minute timeboxed Daily Stand-up meeting to synchronize team members, review progress, and identify issues.
- **Iteration Retrospective** – Is an event held at the end of the iteration for the Agile team to review its practices and identify ways to improve. The retrospective is based on the qualitative and quantitative information presented during the iteration review.
- **Backlog refinement** – Is an event held once or twice during the iteration to refine, review, and estimate stories and enablers in the team backlog.
- **Innovation and Planning (IP) Iteration** – Provides the teams with an opportunity for exploration and innovation, dedicated time for planning, and learning through informal and formal channels. Where a release is on the PI boundary, teams perform final system verification, validation, and documentation. It also provided a planning buffer.

SAFe LEAN AGILE PRINCIPLES

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

Agenda

Why re-invent the wheel when good, well tested ‘ways of working’ (practices/process) already exist.

What are the Essentials needed to provide a way of working ?

- High Level ‘Essential’ SAFe Configuration;
- Two Teams: Program and Team
- Agile Release Trains
- DevOps (CI/CD) – release on Demand
- The Architecture Runway
- Program Level
- Team Level
- The 2 Backlogs
- SAFe principles
- SAFe is continually improving