alok.giri@ncit.edu.np

What is SAFe?

Definition:

SAFe (Scaled Agile Framework) is a **set of organizational and workflow patterns** intended to help enterprises **scale agile**

practices across teams, business units, and portfolios.

Purpose:

- Align **strategy with execution**
- Foster **collaboration across large teams**
- Deliver **value at scale**

Origin and Evolution of SAFe

- **Created by:** Dean Leffingwell
- **First published:** 2011
- Combines principles from:
  - **Agile**
  - **Lean**
  - **Systems thinking**
  - **DevOps**

- Has evolved through multiple versions; latest version (as of 2024): **SAFe 6.0**

Why it matters:

- Designed for **large-scale coordination** across dozens or even hundreds of agile teams

When Should You Consider SAFe?

SAFe is suitable when:

- You have **multiple agile teams** working on interdependent solutions
- You need **alignment between business strategy and technology delivery**
- There's a need for **predictable delivery of value**

- You're in a **regulated or complex industry**

SAFe is not ideal for:

- Small teams or startups
- Organizations not ready for structured change

Introduction to ART (Agile Release Train)

Definition:

An Agile Release Train (ART) is a **long-lived team of agile teams** (typically 50–125 people) that plan, commit, and deliver **together**.

Key Concepts:

- Operates on a **fixed cadence (Program Increment – PI)**, typically 8–12 weeks
- Aligns team objectives with business priorities

- ARTs include all roles required for delivery: Dev, QA, UX, Product, and more

Goal:

Deliver **continuous value** across teams and stakeholders

SAFe Configurations Overview

SAFe is **configurable** to match organizational complexity:

1. Essential SAFe

- The most basic and **foundational** configuration • Includes: Agile Teams, ART, PI Planning, ScrumXP, Kanban

2. Portfolio SAFe

- Adds strategy and funding alignment

- Includes: Lean Portfolio Management (LPM), Strategic Themes

3. Large Solution SAFe

- Supports **large-scale systems** that require multiple ARTs
- Includes: Solution Trains, Solution Architect, Solution Management

4. Full SAFe

- Combines all configurations
- Used in **complex, enterprise-scale** organizations

Four SAFe Configurations

- Essential SAFe: Base level
- Portfolio SAFe: Adds strategic alignment ● Large

Solution SAFe: Adds solution-level coordination ● Full SAFe: Combines all above for enterprise-level agility

Key Roles in SAFe

Role	Primary Responsibility
Release Train Engineer	Facilitates ART operations
Product Management	Defines and prioritizes Features across teams
System Architect	Guides technical vision and enablers
Agile Teams	Cross-functional dev/test/UX teams
Scrum Master	Facilitates team-level agile process
Product Owner (PO)	Owns team backlog, defines stories

Putting It All Together – SAFe in Action

Cycle Example:

1. **PI Planning**: Teams commit to objectives
2. **Iterations**: Agile Teams deliver incrementally
3. **System Demo**: Working software shown at end of each iteration
4. **Inspect & Adapt**: Teams reflect, improve, and re-align

Collaboration Across Levels:

- Portfolio → ART → Agile Teams
- Strategy → Execution → Feedback

Benefits of SAFe

1. Alignment between business and IT
2. Predictable, frequent delivery of value 3.
Transparency and visibility across teams 4.

Improved product quality

5. Cross-team collaboration and shared responsibility

Challenges of SAFe

1. Heavyweight process for smaller teams
2. Requires cultural change and buy-in
3. Complex role structures and terminology
4. Training and onboarding costs
5. Can become bureaucratic if misapplied

When is an Agile Team Considered ‘Large’?

Typical Agile Team Size:

- 5 to 9 members is ideal (as per Scrum Guide)

‘Large’ in Agile Context:

- More than **2-3 agile teams** working on the same product or platform
- More than **10–12 people** trying to coordinate work on related outcomes

Challenges Begin When:

- Teams need **shared coordination**
- Dependencies increase across teams
- **Communication overhead** becomes non-trivial

Why Scaling Agile Is Hard

- Agile thrives on **collaboration and quick feedback**

- Scaling brings risks of:
    - **Misalignment**
    - **Duplicated efforts**
    - **Inconsistent velocity**
    - **Slow decision-making**
- Requires structure **without sacrificing agility**

Coordination Strategies

1. Sync Through Events

- Schedule recurring cross-team meetings:
    - Program Increment (PI) Planning
    - Release Demos

- Inspect and Adapt (I&A) workshops

2. Scrum of Scrums

- Representatives from each team meet regularly
- Share progress, blockers, and dependencies
- Helps with **horizontal alignment**

Coordination Strategies

1. Sync Through Events

- Schedule recurring cross-team meetings:
  - Program Increment (PI) Planning
  - Release Demos
  - Inspect and Adapt (I&A) workshops

2. Scrum of Scrums

● Representatives from each team meet regularly ●
Share progress, blockers, and dependencies
● Helps with **horizontal alignment**

3. Product Owners Sync

● Aligns backlog priorities across teams
● Ensures **product strategy coherence**
● Supports coordinated decision-making on scope tradeoffs

PI Planning (in SAFe)

Definition:

A large-scale planning event (2-day) where all teams on an ART align on:

● What they'll deliver in the upcoming Program Increment (8–12 weeks)
● Risks and dependencies

PI Planning Activities:

- Review vision and roadmap
- Draft team plans
- Identify cross-team dependencies
- Commit to objectives

Benefits:

- Alignment, transparency, shared ownership

Shared Goals and Metrics

Why Shared Goals Matter:

- Prevent isloated efforts
- Foster collaboration
- Prioritize value delivery

Shared Metrics:

- Velocity trends across teams
- Feature and story cycle time ● Deployment frequency
- Defect escape rate
- Objective completion % per PI

Integrating Multiple Agile Teams

Integration is Key for:

- System-level testing
- Avoiding integration hell
- Continuous delivery

Strategies:

- **Define clear interfaces/APIs**
- Test **integration early and often ●**

Use **shared staging environments**

Horizontal and Vertical Team Slicing

Horizontal Slicing (by layers):

- One team owns UI, another owns backend, another owns DB

Issues:

- Tight coupling
- Dependency hell

Vertical Slicing (by features):

- Teams work end-to-end on features
- Promotes autonomy and faster delivery

Preferred approach in Agile scaling

Summary – Key Takeaways

- Large Agile teams need **intentional coordination** and **communication**
- Events like **Scrum of Scrums, PO Syncs, PI Planning** provide structure
- **Shared goals** and metrics drive alignment
- Prefer **vertical slicing** for team structure
- Integration should be **continuous and automated**
- The goal is **scaling without losing agility**