1. SAFe Distilled 2
 1.1 Ch 1: Business Need for SAFe 2
 1.2 Ch 2: SAFe Overview 2
 1.3 Ch 3: Lean-Agile Mindset 6
 1.4 Ch 4: SAFe Principles 7
 1.5 Ch 5: Lean-Agile Leaders 10
 1.6 Ch 6: The Agile Release Train 12
 1.7 Ch 7: Planning a Program Increment 13
 1.8 Ch 8: Iterating 17
 1.9 Ch 9: Executing the Program Increment 19
 1.10 Ch 10: Inspect and Adapt 23
 1.11 Ch 11: Large Solution SAFe Overview 26
 1.12 Ch 12: Defining Large and Complex Solutions 27
 1.13 Ch 13: Solution Train Execution 31
 1.14 Ch 14: Lean Portfolio Management 32
 1.15 Ch 15: Strategy and Investment Funding 34
 1.16 Ch 16: Agile Portfolio Operations 35
 1.17 Ch 17: Lean Governance 36
 1.18 Ch 18: The Guiding Coalition 37
 1.19 Ch 19: Designing the Implementation 38
 1.20 Ch 20: Implementing Agile Release Trains 39
 1.21 Ch 22: Sustaining and Improving 43
 1.22 Ch 21: Launching More ARTs and Value Streams 44

SAFe Distilled

The following are summary notes for the book *SAFe 4.5 Distilled: Applying the Scaled Agile Framework for Lean Enterprises (2nd Edition)*.

- Ch 1: Business Need for SAFe
- Ch 2: SAFe Overview
- Ch 3: Lean-Agile Mindset
- Ch 4: SAFe Principles
- Ch 5: Lean-Agile Leaders
- Ch 6: The Agile Release Train
- Ch 7: Planning a Program Increment
- Ch 8: Iterating
- Ch 9: Executing the Program Increment
- Ch 10: Inspect and Adapt
- Ch 11: Large Solution SAFe Overview
- Ch 12: Defining Large and Complex Solutions
- Ch 13: Solution Train Execution
- Ch 14: Lean Portfolio Management
- Ch 15: Strategy and Investment Funding
- Ch 16: Agile Portfolio Operations
- Ch 17: Lean Governance
- Ch 18: The Guiding Coalition
- Ch 19: Designing the Implementation
- Ch 20: Implementing Agile Release Trains
- Ch 22: Sustaining and Improving
- Ch 21: Launching More ARTs and Value Streams

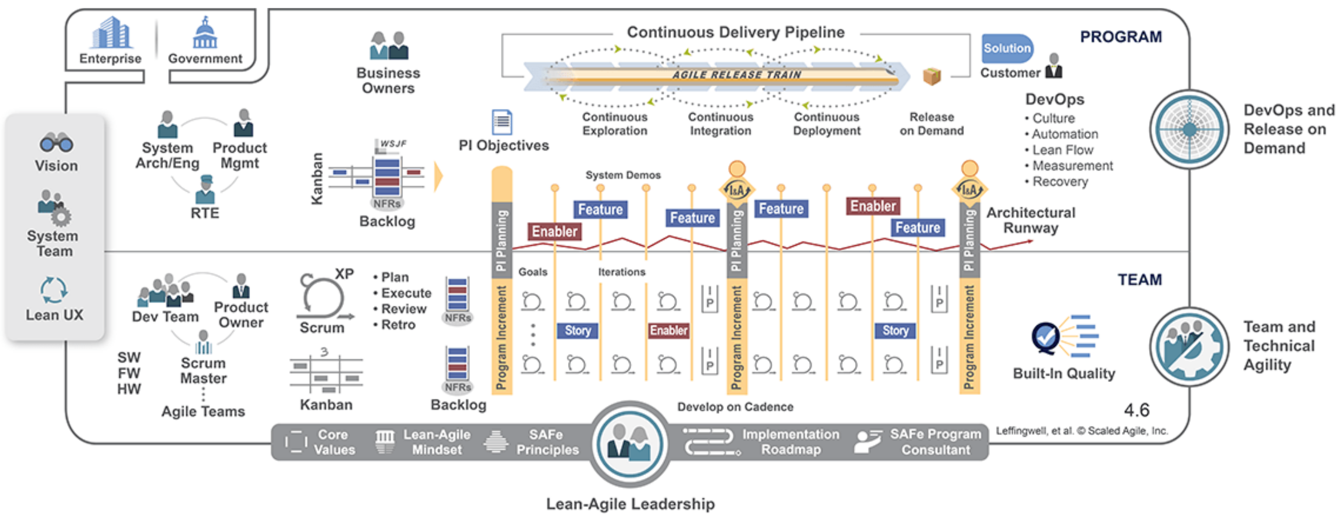
Ch 1: Business Need for SAFe

- No company can make, deliver, or market its product efficiently without technology
- **Adaptation** is essential, or you'll go extinct
- Challenges… maintaining existing things, defense against attackers, theft of IP, bigger, more complicated, more integrated, bigger impact, failure has broader consequences, tech skills keep expanding
- Outsourcing challenges… delayed communication, dependency on others, loss of internal capabilities, quality, regulatory compliance
- W.E. Deming said the problems are usually **system** problems, not poor performance of people
- Previous systems were built for **control and stability**, not innovation and speed
- New bodies of knowledge
  - Agile development – lightweight development processes; manifesto defined the philosophy
  - Systems thinking – set of interrelated elements; two systems (one for the customer's benefit, the other constituting the org that builds it)
  - Lean product development – Lean thinking (Toyota Production System and Total Quality Management) + product development flow; principles… eliminate waste/delays, maximize customer value, reduce lead time, high quality, sustainable flow (batch size, WIP limits, etc.), respect, using MVPs, continuous improvement (kaizen), top-down buy-in
- Benefits
  - Quality – defect rates, expenses caused by flawed products, client dissatisfaction
  - Productivity
  - Employee engagement
  - Faster time-to-market
  - Program execution – predictability, on-time delivery
  - Alignment
  - Transparency

Ch 2: SAFe Overview

The purpose is to align, collaborate, and deliver over across multiple teams.

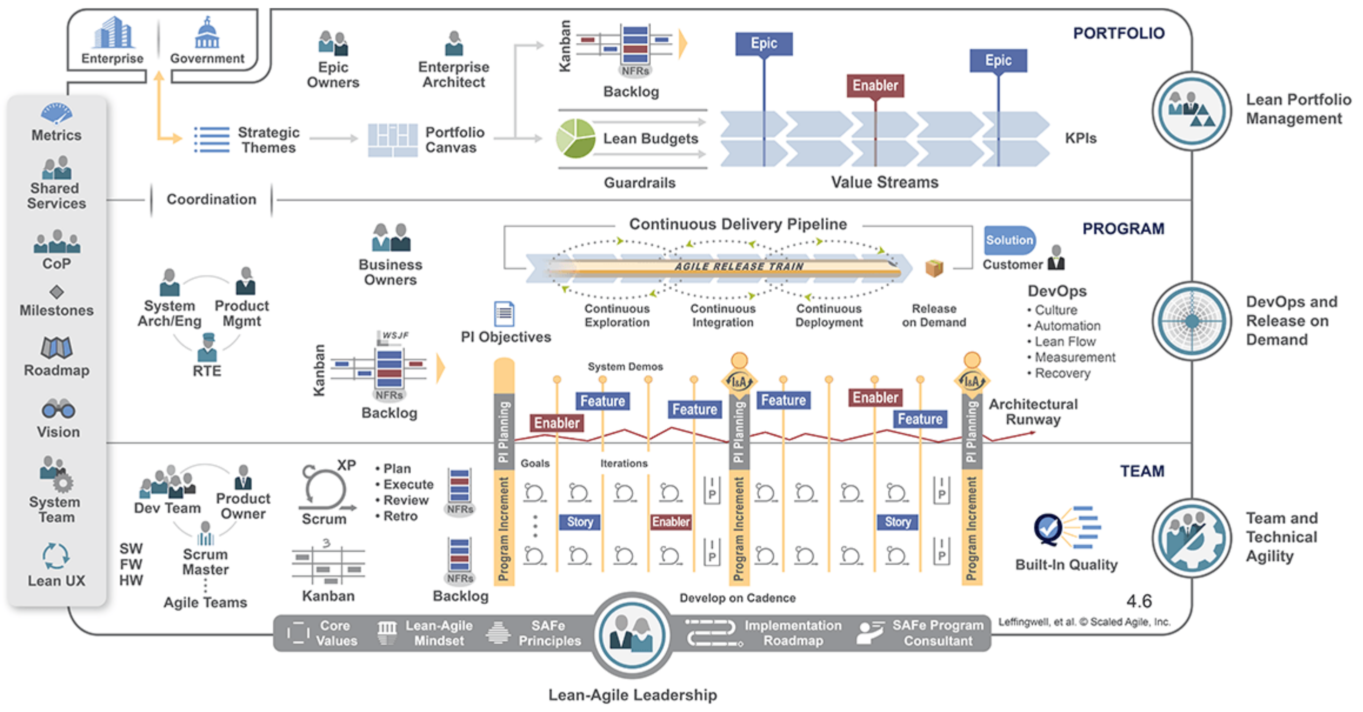
Essential SAFe



(Source: https://www.scaledagileframework.com/essential-safe/)

- Agile Release Train (ART) – multiple Agile teams, key stakeholders together to pursue an ongoing solution mission (shared vision, roadmap, and backlog); delivers features (things users want) and enablers (technical infrastructure) to do this
- Team iterations are synchronized by start/end dates (usually two weeks) to deliver finished increments
- Program Increment – longer, fixed timeboxes for planning, execution, inspection, adaptation
- Need a continuous delivery pipeline; DevOps helps to plan, develop, test, deploy, release, and maintain the system
- Key roles
  - System Architect – define the overall architecture, non-functional requirements (NFRs), critical subsystems; uses systems thinking
  - Product Manager – internal voice of the customer, owns the program backlog (features and enablers); bridge between customers and Product Owners
  - Release Train Engineer – chief Scrum Master for the ART; sets up practices, planning, etc.
  - Business Owners – primary business expertise, know about compliance, ROI
  - Customers – deciders of value
- Essential elements
  - Lean-agile principles
  - Real agile teams and trains
  - Cadence and synchronization
  - Program increment planning
  - DevOps and releasability
  - System demo
  - Inspect and adapt (I&A)
  - Innovation and planning (IP) iteration
  - Architectural runway
  - Lean-agile leadership (leaders actively participate and take responsibility for the implementation)

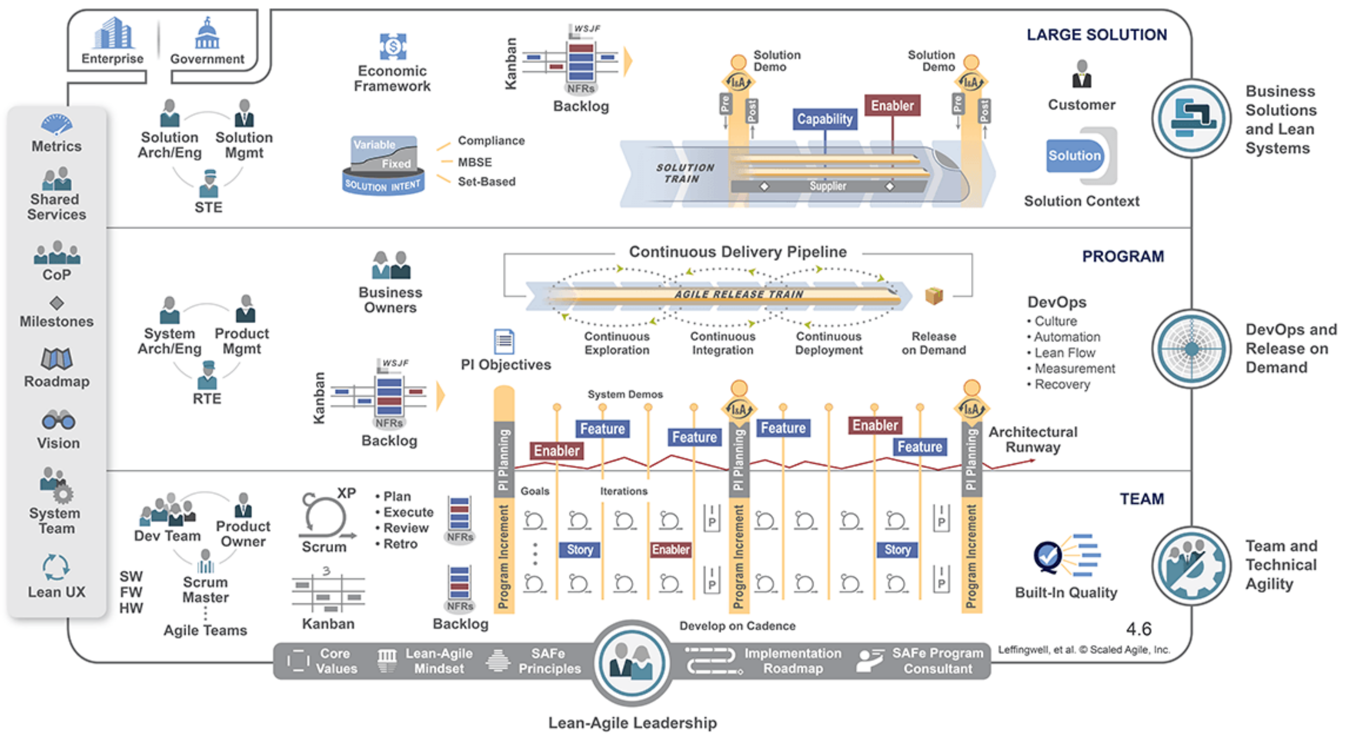
Portfolio SAFe



(Source: https://www.scaledagileframework.com/portfolio-level/)

- Additional elements
    - Lean budgets – fast, empowered decision-making
    - Value streams – series of steps used to build product
    - Portfolio kanban
- Key roles
    - Lean Portfolio Manager – high-level decisions, strategy, investment funding, Agile operations, Lean governance
    - Epic Owner – coordinate portfolio epics
    - Enterprise Architect – person or team that works across teams

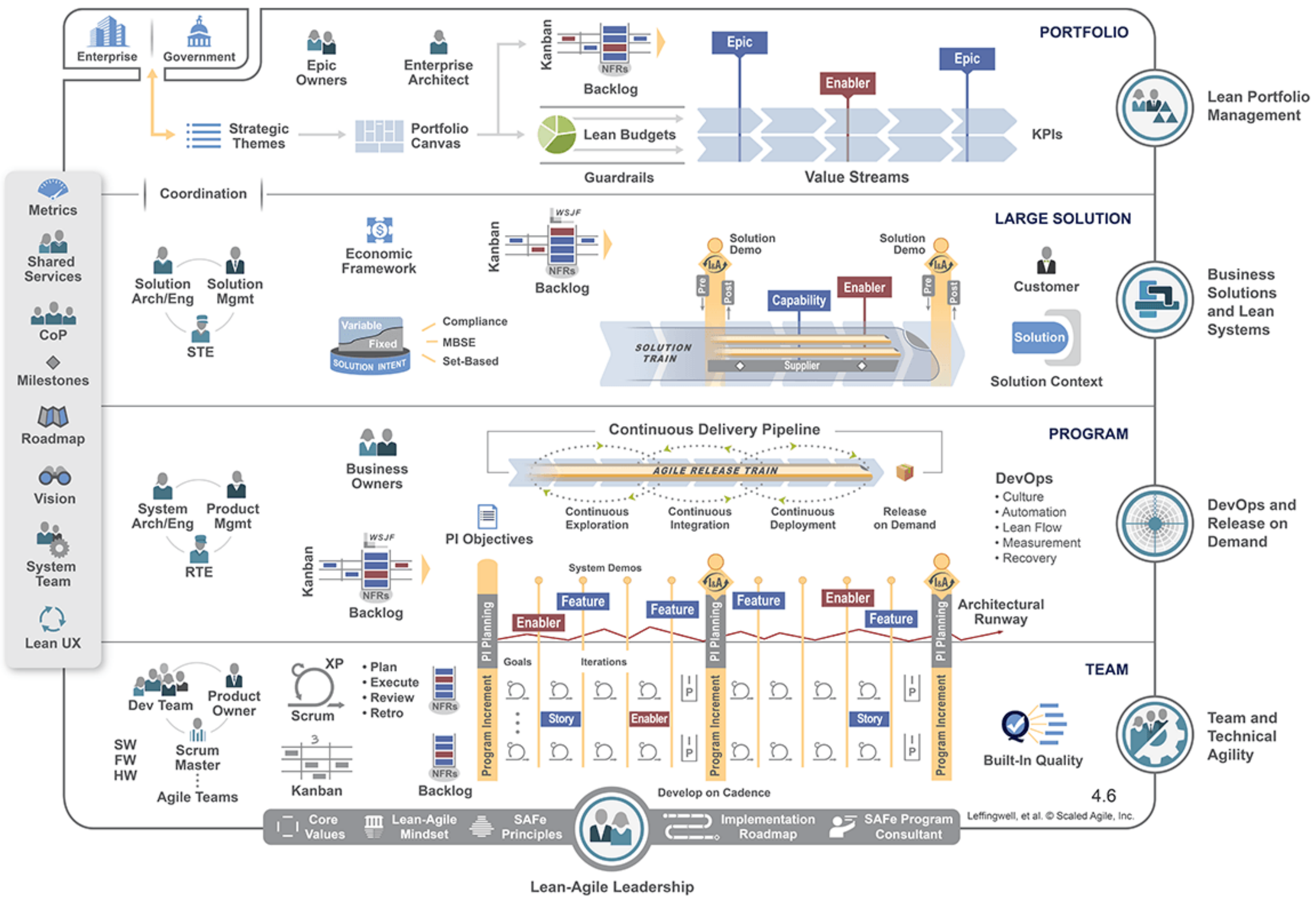
Large Solution SAFe



(Source: https://www.scaledagileframework.com/large-solution-level/)

- Additional elements
  - Solution train – aligns people and work with a common solution vision, mission, and backlog
  - Supplier – development of components and subsystems
  - Economic framework – financial boundaries
  - Solution context – how the system is packaged/deployed
  - Solution Kanban – flow of capabilities and enablers
- Key roles
  - Solution Architect/Engineer – person or team that sets technical vision for the solution
  - Solution Management – creates solution vision, backlog, roadmap, defines requirements, guides work through solution Kanban
  - Solution Train Engineer – facilitates work of all ARTs and suppliers

Full SAFe



(Source: https://www.scaledagileframework.com/#)

This level combines all the others and allows multiple SAFe configurations.

Spanning Palette (what all versions have)

- Metrics – measure throughout and at PI planning
- Shared Services – specialty roles that can't be dedicated full time to any train
- Community of Practice (CoP) – group of experts who have some practical knowledge
- Milestones – fixed date, program increment, learning milestones
- Roadmap – planned deliverables on a timeline
- Vision – future view of the solution (customer needs, features, capabilities)
- System Team – builds the Agile dev environment, CI, test automation

The Foundation

- Lean-agile leaders – lead and train others in ways of thinking; lifelong learners, apply and embrace principles and practices
- Core values
  - Alignment – everyone knows the strategy and where they fit
  - Quality – poor quality doesn't scale; you need customer satisfaction and predictable delivery
  - Transparency – builds trust, which leads to performance, innovation, risk-taking, relentless improvement
  - Execution – minimal overhead to create stable, long-lived teams
- Lean-agile mindset – beliefs, assumptions, actions of leaders
- SAFe principles – (coming later in the book)
- SAFe implementation roadmap – how to implement SAFe
- SAFe program consultants – change agents who know SAFe and want to improve systems development

Ch 3: Lean-Agile Mindset

Thinking Lean

- Goal: Value
  - Deliver max value in the shortest sustainable lead time
  - High quality to customers and society
  - Other goals... high morale, emotional safety, physical safety, customer delight
- Respect for People and Culture
  - People do the actual work, not the system
  - Learn problem-solving skills and reflection skills
  - Not just employees... suppliers, partners, customers, community
- Flow
  - Continuous flow of incremental value delivery based on feedback and tuning
  - Understand the full value stream
  - Visualizing and limiting WIP
  - Reduce batch sizes
  - Manage queue lengths
  - Focus on reducing delays and eliminating waste
- Innovation
  - Gemba walks around where the work is happening; "No useful improvement was ever invented at a desk." (Taiichi Ohno)
  - Provide regular time/space for people to be creative
  - Avoid the tyranny of the urgent. You can't innovate if 100% on firefighting
  - Apply innovation accounting to see if things are working
  - Validate new ideas; pivot without mercy or guilt
- Relentless Improvement
  - Optimize for the whole org, not just the dev parts
  - Consider facts carefully, then act quickly
  - Find root causes and apply countermeasures
  - Reflect during milestones
- Foundation: Leadership
  - Top management owns the process
  - Train them to lead by example

Embracing Agility

- Teams should be self-managing and self-organizing
- Need authority and autonomy to plan, execute, decide, adapt
- Teams monitor their own progress, solve problems together, improve their processes
- Although independent, they must align with management on business goals and standards
- Values; both sides are important
  - Individuals and interactions > processes and tools
    - "If you can't describe what you are doing as a process, then you don't know what you are doing." (Deming) This means processes like Scrum, Kanban, etc. do matter. Tools like chat, wikis, etc. do matter.
  - Working software > comprehensive documentation
    - Only document what's necessary; no documentation for its own sake
  - Customer collaboration > contract negotiation
    - Continuous feedback from the customer
    - Win/lose contracts lead to poor economic outcomes and distrust
  - Responding to change > following a plan
    - There is a plan, but change is inevitable
- There is no end-state for finding better ways

Ch 4: SAFe Principles

Principle 1: Take an Economic View

- Deliver incrementally, early, and often
  - Customers get continually growing value instead of "big bang" value at the end with waterfall
  - Mitigates risk by getting feedback more often
- Sequence jobs for maximum benefit

- **Reprioritize work continuously** depending on economic/technical facts known at the time
- **Weighted Shortest Job First (WSJF)** means picking the next task that delivers the most value in the shortest amount of time
    - WSJF = cost of delay / job size
    - Cost of delay = user or business value + time criticality + risk reduction or opportunity enablement
    - Don't need $$ values for these, as those can take too much overhead to compute; make them relative
    - Job **duration** is harder to find (how long will it take to build something you've never built), which is why relative size is usually best

Feature	User or business value	Time criticality	Risk reduction -or- opportunity enablement	Cost of delay	Job size	Weighted shortest job first
	+	+	=	/	=	
	+	+	=	/	=	

- Scale for each parameter: 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 (e.g., some feature has the lowest cost of delay or lowest job size)
- The highest priority is the highest WSJF

Principle #2: Apply Systems Thinking

- Deming: “A system must be managed.”
- The solution (e.g., website, medical device) is a system
    - Know what’s in it, where the boundaries are, how it interacts with other things
    - Optimizing the part does not necessarily optimize the whole
    - Intentional design is fundamental
    - The value of a system passes through its interconnections
    - A system can evolve no faster than its slowest integration point
- The org is a system, too
    - If you don’t manage this, subsystems will optimize greedily causing problems elsewhere
    - Building complex systems is social. Leaders must create environments for collaboration to happen.
    - Treat suppliers and customers as partners to build trust
- Understand the full value stream
    - This is the only way to reduce the total time from concept to cash
    - Value stream mapping is the tool
- Only management can change the system
    - Systematic problem solving
    - Take the long view
    - Proactively eliminate impediments
    - Lead org changes

Principle #3: Assume Variability, Preserve Options

- Point-based design (most common): pick one solution/technology quickly and modify the design until the system is built
- Set-based design: pick several flexible options, eliminate weaker options over time
- (Geoff’s distillation: consider many options, choose based on flexibility/adaptability because change is inevitable.)

Principle #4: Build Incrementally with Fast, Integrated Learning Cycles

- Not well defined in the book…
- **Integration points** could mean seams where different components connect – database, API, third-party service – where stuff goes in and out of a component.
- It could also mean integrating changes into the entire system. If you can’t change quickly, you can’t learn quickly either.
- It could also be the connections between components in your larger system. Example: job that creates data, another job that takes that data to do something else

Principle #5: Base Milestones on Objective Evaluation of Working Systems

- Ship smaller batches of working software more often.
- Milestone = program increment
- Re-evaluate at each PI what you need to do next based on what you've learned from your working system

Principle #6: Visualize and LImit WIP, Reduce Batch Sizes, and Manage Queue Lengths

- Too much WIP = multitasking and context switching, overloads people, reduces focus, productivity, and throughput. Visualize it with a Kanban board
- If the batch size is too small, you have higher **transaction costs** (planning, implementing, testing). Too high and you're **holding costs** (value delivered because you shipped) will be high.
- Queue lengths
  - Long queues are bad -- longer cycle times, increased risk, increased variability, lower motivation
  - Little's law -- average wait time = queue length / processing rate
  - To reduce wait time, (1) shorten the queue, (2) process faster. The second option can only go so far without affecting quality or burning out.

Principle #7: Apply Cadence; Synchronize with Cross-Domain Planning

- "Solution development is an inherently uncertain process. This uncertainty conflicts with the business's need to manage investment, track progress, and plan and commit to a longer-term course of action."
- Cadence
  - Makes waiting times for new work predictable
  - Supports regular planning and cross-functional coordination
  - Limits batch sizes to a single interval
  - Controls injection of new work
  - Provides scheduled integration points
- Synchronization
  - Facilitates cross-functional trade-offs of people and scope
  - Aligns all stakeholders
  - Provides for routine dependency management
  - Supports integration and assessment of full system
  - Provides feedback from multiple perspectives
- Typically implemented through 2-week sprints, 8-week PIs
- PI planning is essential for cross-functional work
  - Assess the current state of the solution
  - Realign all stakeholders to a common technical and business vision
  - Plan and commit teams to the next PI

Principle #8: Unlock the Intrinsic Motivation of Knowledge Workers

- "Knowledge workers are people who know more about the work they perform than their bosses." -- Peter Drucker
- Managers need to unlock the intrinsic motivation of knowledge workers
- **Leverage systems thinking** -- communicate across functional boundaries, make decisions based on economics, receive fast feedback about the viability of their solutions, continuous learning/mastery
- **Understand the role of compensation** -- too much money, threats, intimidation, and fear are at odds with ideation, innovation, and engagement
- **Create an environment of mutual influence** -- disagree when appropriate, advocate for positions you believe in, make needs clear and push to achieve them, enter into joint problem solving with management and peers, negotiate/compromise/agree/commit
- **Provide autonomy, mastery, and purpose --** opt toward self-direction, people want to grow in their careers, connect the work to the enterprise

Principle #9: Decentralize Decision-Making

- Complete centralized decision-making is a bottleneck, and lowers empowerment
- Centralize
  - Infrequent decisions (e.g., product strategy)
  - Long-lasting (e.g., tech platform)
  - Significant economies of scale (e.g., standard tooling across teams)
- Decentralize everything else

    - Frequent (e.g., backlog priorities for defects)
    - Time-critical (e.g., hotfixes)
    - Local information (e.g., resolution of a design problem)

Ch 5: Lean-Agile Leaders

Only the enterprise's executives, leaders, and managers can change an continuously improve the system in which people work

Exhibit the Lean-Agile Mindset

If you have words but no actions, it won't work. Leaders must embrace the Ch 3: Lean-Agile Mindset (value, people/culture, flow, innovation, relentless improvement, agility).

Know the Way and Emphasize Lifelong Learning

(The book recommends leaders take a 2-day course "Leading SAFe" (https://www.scaledagile.com/certification/courses/leading-safe/) to help implement it. It's about $1000.)

Book recommendations: https://www.scaledagileframework.com/recommended-reading

Other ways leaders can facilitate learning:

- Sponsor/participate in book clubs
- Host lunch-and-learns
- Benchmark with other companies
- Support outside conferences and educational opportunities

Develop People

(From *Managing for Excellence* by Bradford and Cohen)

Leader as Expert

Characteristics	Challenges
• Technician, master craftsman • Promoted because they were best at their job • Problem solver, the one with the answers • Understands domain and tech • Works when people leave them alone	• Limits learning and growth of direct reports • Focuses on tech to the detriment of human factors • Knowledge becomes outdated

Leader as Conductor

Characteristics	Challenges
• Central decision maker, nerve center, coordinator • Orchestrates individual parts of the org into a harmonious whole • Subtle and indirect manipulation to their solution • Manages across individuals, teams, depts. • Works by coordinating others	• Narrows the focus of direct reports to their own areas • Pushes conflict upward, looking for the boss to fix it • Uses systems and procedures to control work • Works harder and harder without realizing full potential

Leader as Developer of People

Behaviors	Benefits

• Creates a team jointly responsible for success • Asks, “How can each problem be solved in a way that further develops my people’s commitment and capabilities?” • Gives credit to the team for success, shoulders responsibility when things go wrong • Shows empathy and support when the team makes mistakes • Creates a learning culture • Fosters an environment that rewards risk-taking and innovation without fear • Works by developing others' abilities	• Increased direct report ownership and responsibility • Increased employee engagement and motivation • Allows leader to spend more time managing laterally and upward • No limit to the power of getting things done

Inspire and Align with Mission

Define the mission, and reduce boundaries and conditions for teams to address it. This is the *what* and the *why* (not the *how*, that’s for the team to decide).

Eliminate demotivating polices/procedures that promote unhealthy competition, encourage favoritism, or cause busywork.

Decentralize Decision-Making

This is Principle 9: https://lirio-llc.atlassian.net/wiki/spaces/~174556444/pages/375717937/Ch+4+SAFe+Principles#Principle-%239%3A-Decentralize-Decision-Making

Unlock the Intrinsic Motivation of Knowledge Workers

This is Principle 8: https://lirio-llc.atlassian.net/wiki/spaces/~174556444/pages/375717937/Ch+4+SAFe+Principles#Principle-%238%3A-Unlock-the-Intrinsic-Motivation-of-Knowledge-Workers

Evolve the Development Manager Role

Goal of SAFe: nearly autonomous, cross-functional teams and Agile Release Trains (how stuff gets shipped). Lean infrastructure, empowered teams to do local decision making.

You still need people managers, though.

- Recruiting/retaining talent
- Vision and mission alignment
- Support built-in quality and Agile engineering processes
- Coaching Agile teams
- Providing transparency
- Serving as business owners

More details: https://www.scaledagileframework.com/lean-agile-leadership/

Adopt a Servant-Leadership Approach

- Listening
- Empathy
- Self-awareness
- Persuasion (instead of authority)
- Conceptualization (the *why*)
- Stewardship
- Commitment to the growth of people

Partnering with HR

More details: https://www.scaledagileframework.com/agile-hr/

- Embrace the new talent contract
- Foster continuous engagement
- Hire for attitude and cultural fit
- Move to iterative performance feedback

- Take the issue of money off the table
- Support meaningful learning and growth

On the Future of Leadership

- Leadership is a task, not a permanent identity
- Hyper-transparency of a wide range of information fosters effective leadership and results
- Use non-monetary incentives instead of financial rewards

Note: Not sure if this is still in vogue, given these points are from 2008 (https://hbr.org/2008/05/leaderships-online-labs)

Ch 6: The Agile Release Train

Overview

- **Agile Release Train**
  - long-lived team of teams that develops and delivers solutions incrementally
  - organized around value streams
  - <u>Goal</u>: achieving continuous flow of value
  - Define new functionality Implement Acceptance Test Deploy (repeat)
- Principles
  - Fixed schedule (e.g., PI = 8 weeks), fixed iterations (e.g., sprint = 2 weeks)
  - Predictable estimate of how much cargo can be delivered each PI
  - Most people dedicated full-time
  - Dedicate time for innovation and planning activities
  - Uses DevOps and Lean UX to get fast feedback and reduce waste

ART Organization

- Cross-functional instead of silos
- Each team has the skills to effectively deliver a feature with a minimum dependency on others
- Goal: faster flow of value with minimum overhead
- Roles
  - **Release Train Engineer** – servant leader and coach, facilitates ART events, communicates with stakeholders, escalates impediments, helps manage risk, drives relentless improvement
  - **Product Management** – owns what gets built based on vision, roadmap, and current backlog; collaborates with customers and POs to understand needs and validate solutions
  - **System Architect** – defines technical/architectural vision; defines major components, interfaces, non-functional requirements
  - **Business Owner** – has business and technical responsibility for governance, compliance, and ROI for the solution; evaluates solution's business value and fitness for use
  - **Customer** – buys the solution, participates in solution development
  - **System Team** – builds the dev environment (CI/CD, testing)
  - **Shared Services** – part-time specialists (e.g., data security, information architects, DBAs, tech writers)

Develop on Cadence. Release on Demand.

- If you work independently on different schedules with different priorities, it's difficult to integrate the system routinely
- Continuous delivery is ideal, but there may be reasons you can't do that

Vision

- Product communicates the intent and direction of the solution (e.g., what does it do? what problems does it solve and for whom?)
- PI planning involves presenting the vision to get people excited and aligned

Features

- **Feature** – larger system behavior that fulfills the user's needs
- **Benefit hypothesis** – proposed measurable benefit (e.g., reduced planned downtime)

- Acceptance criteria – tells you how to validate the hypothesis; drives the stories and tests

Program Backlog

- Product identifies, prioritizes, and sequences features
- Originated from anywhere – customer, Product, PO, architect, etc.
- **Enabler** – work that gives you more runway
    - Exploration – research/prototype to understand customer needs, explore prospective solutions, and evaluate alternatives
    - Architecture – focused on smoother and faster development
    - Infrastructure – build, enhance, and automate the development, testing, and deployment environments
    - Compliance – compliance-based activities
- SMEs come together to size features so they can be prioritized by weighted shortest job first (WSJF)

Roadmap

- Consists of anticipated features and other milestones
- Owned by Product

Agile Teams Power the Train

- People that define, build, test, and deploy
- Follows Agile practices like Scrum, XP, Kanban
- Estimates and manages its own work
- Determines technical design in its area of concern
- Commits to the work it can accomplish each iteration/PI
- Implements and tests functionality and promotes work to other environments
- Supports and/or builds the automation needed to implement the CD pipeline
- Continually improves the process and deliverables
- Roles
    - **Scrum Master** – servant leader, facilitates meetings, fosters Agile behavior, helps remove impediments, interacts with the larger org, helps the team maintain focus, builds a high-performing and self-managing team
    - **Product Owner** – owns the team backlog, defines stories, acts as the customer, prioritizes work, works with Product to plan PIs
    - **Team** – individual contributors that do the work, cross-functional

User Stories and the Team Backlog

- Not requirements; short, simple descriptions of a small piece of desired functionality told from the user's perspective
- Written in language that explains the intent to business and technical people
- "As a <role> I can <activity> so that <business value>"
- Ideally acceptance tests line up with the acceptance criteria
- Stories can also be enablers (see Program Backlog section above)
- Estimating
    - **Story points** capture volume, complexity, knowledge, uncertainty
    - Every story is relative to the smallest story (1 SP); use 1, 2, 3, 5, 8, 13, 20, 40, 100
    - Techniques: planning poker, white elephant sizing
    - **Velocity** – number of points per sprint the team can achieve
    - Velocity varies per team and for the type of work. ART's velocity is the sum of all teams
- Team backlog
    - It contains all things the team could do (make work visible)
    - Owned by the PO
    - Inputs
        - Program backlog
        - Team (e.g., refactor, maintenance, tech debt)
        - Other stakeholders (e.g., team dependencies, other commitments, spikes/research)
- Capacity allocation
    - Determine how much work should be applied to user stories, refactors, and maintenance

Ch 7: Planning a Program Increment

Overview

- PI planning is a cornerstone event; it's about alignment
    - Teams come together to define and design the system that best fulfills the ART's vision
    - Commit to near-term PI objectives
    - Sense of shared mission, responsibility, cooperation, and collaboration
    - Responsibility of planning moves from a central authority to the teams who do the work
- Usually facilitated by the Release Train Engineer

Preparation for the PI Planning Event

Organizational Readiness

- **Planning scope and context** – do you understand the product, system, or tech domain?
- **Business alignment** – is there agreement on priorities among the Business Owners?
- **Agile teams** – does each team have the resources (dev, testers, Scrum Master, PO)?

(See Chapter 20 – Implementing Agile Release Trains)

Facility Readiness

- **Facility** – need a big enough space, breakout rooms
- **Technical and communications support** – people to help before and during the event
- **Communication channels** – consider how remote attendees will participate

Content Readiness

- **Executive briefing** – senior exec presents the business context
- **Product/solution vision briefing** – Product presents the vision, "top 10 features" in the Program Backlog
- **Architecture vision briefing** – CTO communicates architectural strategy, new enablers, and non-functional requirements

The Role of the Facilitator

- Organizes the agenda to ensure people are participating and to keep discussions on track
- (SAFe has a two-day planning agenda, which is discussed below)

Day 1: Create and Review Draft Plans

Business Context (1 hr)

- Senior exec sets the tone for PI planning by talking about things like performance, strategy, SWOT analysis, customer satisfaction, org developments, operating plans, etc.
- Rally the troops around challenges and opportunities, drive motivation and enthusiasm for the PI and the evolving solution

Product/Solution Vision (1.5 hrs)

- Product Manager presents current vision and objectives for the PI, priorities

Architecture Vision and Development Practices (1 hr)

- System Architect presents vision for the architecture (new epics for common infrastructure, large-scale refactors, system level NFRs)
- Announce any changes to standard development practices, new tools, techniques, built-in quality practices, build pipelines

Team Breakouts (3 hrs)

- Teams draft initial plans to grasp scope of priorities necessary for development, resolve dependencies, and understand the potential reuse for common code.
- “It’s an intense and active time.”
- All teams have a list of the PI objectives/priorities
- Physical:
  - flip chart paper for each sprint, one for risks, one for IP iteration (we don’t currently do this: https://www.scaledagileframework.com/innovation-and-planning-iteration/)
  - stickies for user stories
  - stickies for maintenance work
  - stickies for exploration enablers
  - stickies for infrastructure enablers
  - stickies for risks & dependencies
- Capacity-based planning
  - New teams: start with 8 SPs per sprint
  - 1 SP = a story that can be completed end-to-end in one day; all other stories are relative to this one
  - Any story larger than 8 SP should be split
  - (I disagree with the approach that you make everyone use story points. The book says, “…which is vital for feature and epic-level estimating and conversion into cost estimates where necessary.” This means SPs are translated into time, which the originator of the SP said was precisely the opposite of what he intended.)
- Hourly Planning Checkpoints
  - (The book has a “PI Planning Radiator” worksheet where the RTE checks in with each team every hour to verify that specific things have been done.)
  - (This seems like considerable overhead for everyone involved. Maybe have a shared Excel file that the teams fill in real-time that the RTE can watch?)

Draft Plan Review (1 hr)

- ART gets together to review each team’s draft plans
- Teams give a readout (strict time box)
- Business Owners are present
- Agenda
  - Velocity and load (number of points committed to do this PI)
  - Draft PI objectives
  - Program risks & impediments
  - Q&A
- Most of the teams can call it a day after this

Management Review and Problem Solving

- Who’s here: RTE, Product Manager, System Architect, Business Owners, Scrum Masters, Product Owners, SMEs
- Agenda
  - What did we just learn?
  - Where do we need to adjust vision, scope, or resources?
  - Where are the bottlenecks?
  - What features must be de-scoped?
  - What decisions must we make between now and tomorrow to address these issues?

Day 2: Finalize Plans and Commit

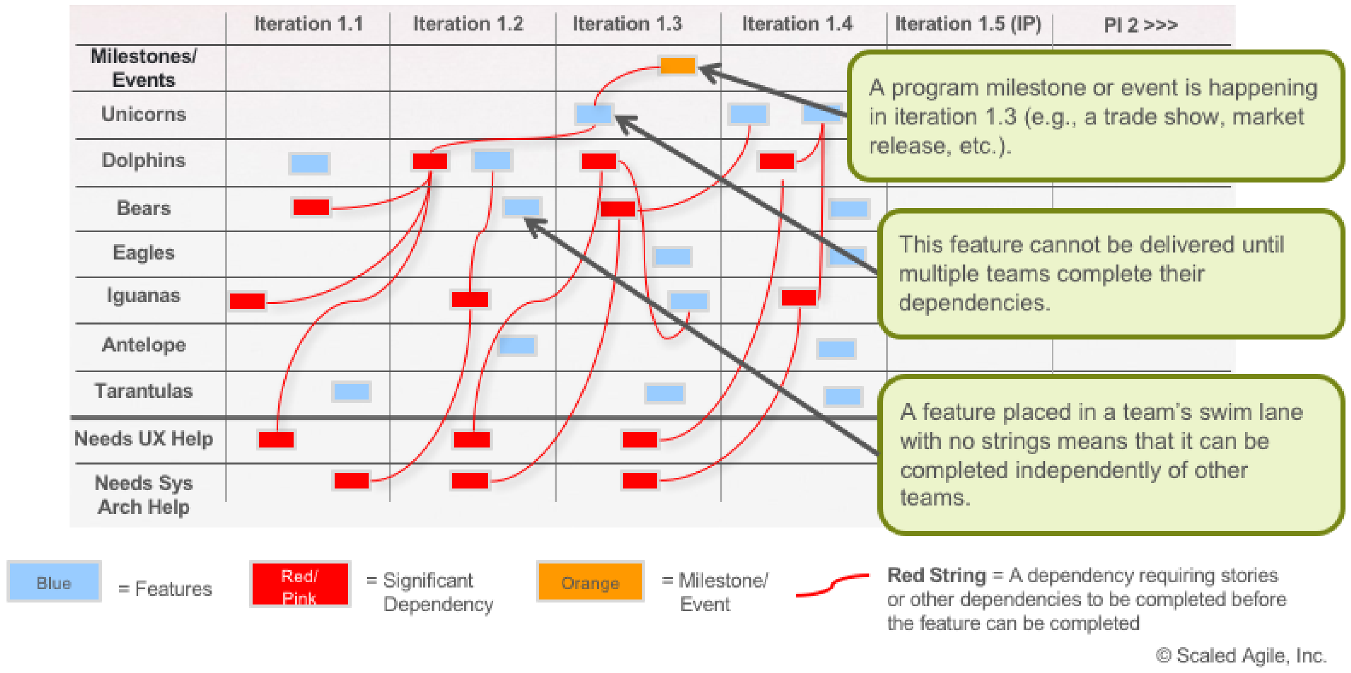
Planning Adjustments (1 hr)

- Read out what was discussed in the “Management Review and Problem Solving” meeting from yesterday

Team Breakouts (2 hrs)

- Based on new knowledge, teams finalize plans
- Business Owners assign business value to PI objectives from 1 to 10 (Why wait to do this? Shouldn’t this have been clarified from the start?)
- Program board is updated with all features and cross-team dependencies

- Consolidate program risks, impediments, and dependencies
- RTEs check in hourly for progress
- Team PI objectives: The team explains to the Business Owner (concisely and in terms business understands) the essence of the value implementing this set of features would accomplish.
- Stretch objectives: work not officially planned; nice-to-haves or things that are more nebulous
- Establish business value: know the percentage of business value delivered in the PI
- Program Board (I think this would be a nightmare to implement for Lirio. Everything would be dependent! A Gantt chart would probably work better. The book has a vignette from the originator of this board: Its purpose is to visualize dependencies, which usually slow things down.)


Iteration 1.1
Iteration 1.2
Iteration 1.3
Iteration 1.4
Iteration 1.5 (IP)
PI 2 >>>
Milestones/ Events
Unicorns
Dolphins
Bears
Eagles
Iguanas
Antelope
Tarantulas
Needs UX Help
Needs Sys Arch Help
A program milestone or event is happening in iteration 1.3 (e.g., a trade show, market release, etc.).
This feature cannot be delivered until multiple teams complete their dependencies.
A feature placed in a team's swim lane with no strings means that it can be completed independently of other teams.
Blue = Features
Red/ Pink = Significant Dependency
Orange = Milestone/ Event
Red String = A dependency requiring stories or other dependencies to be completed before the feature can be completed
© Scaled Agile, Inc.

Final Plan Review (2 hrs)

- Each team talks about
  - Changes to velocity/load
  - Final PI objectives with business value
  - Program risks & impediments
  - Q&A

Addressing Program Risks and Impediments (1 hr)

- (The book doesn't say how to do this. I guess it's the same audience as the management review.)
- ROAM the risks
  - Resolved – no longer a risk
  - Owned – can't fix it but someone has the ball
  - Accepted – we can't avoid it so we'll deal with it when it happens
  - Mitigated – reduced impact or delayed

The Commitment (15 mins)

- Ask each team how confident they are in completing their work; fist-vote (1 finger = low, 5 fingers = high)
- Looking for an average of 3
- Management creates a culture that risk-taking and commitment are the norm
- Teams commit to do everything reasonable to meet the objectives
- If the team learns it can't meet the objectives, escalate ASAP

Planning Retrospective

- Get feedback on how the PI went
- Identify next actions and who owns them

Moving Forward and Final Instructions to Teams

- RTE and Product syncs up with the teams to affirm team PI objectives
- (Hasn’t this already occurred, though?)

Ch 8: Iterating

Overview

- Now that the PI is planned, it’s time to start building/delivering
- Usually 2 weeks
- Goals:
    - Provide a regular, predictable cadence for teams to product an increment of value
    - Refine those items previously developed

The Iteration Cycle

This is where we execute PDCA (plan, do, check, adjust). This section covers textbook Scrum.

Planning

- Involves refining the details and adjusting the initial iteration plans created during PI planning
- Members: PO, Scrum Master, Dev Team
- Time: max of 4 hours
- Inputs
    - Team and program PI objectives
    - Stories from PI planning
    - Existing backlog stories
- Steps
    - Calculate team capacity
    - Discuss each story with the PO; get acceptance criteria and estimate
        - Look at high priority stories first
        - Talk about implementation options, technical issues, NFRs, dependencies
    - Keep planning until no capacity left
    - Create and agree on iteration goals
    - Commit
- Outputs
    - Backlog with refined and prioritized stories for this iteration
    - Iteration goals
    - Commitment
        - Do everything you say you’d do -or- immediately raise a red flag
        - Too much: burnout, inflexibility, quality problems
        - Too little: unpredictability, lack of focus on results
        - Business commits to not change priorities during the iteration
- Think about how you’ll demo during iteration review

Execution

- Goal: Completely **define**, **build**, and **test** multiple stories each iteration
- Use **vertical slices** to deliver small increments of working functionality across all architectural layers

Tracking Iteration Progress

- Scrum board (a.k.a. Big Visible Information Radiator)
- Scrum Master role
    - Facilitating team events

    - Fostering backlog refinement throughout iteration and PI
    - Encouraging the team to sound an alarm when iteration or PI objectives may be at risk
    - Communicating to/from Scrum of Scrums and PO Sync
    - Fostering the use of Agile software engineering practices
    - Ensuring defects aren't pushed to the next iteration
    - Helping with PI planning
    - Supporting deployment/release activities

Daily Standup

- Format
    - What did I work on yesterday
    - What will I do today
    - What impediments might prevent us from meeting iteration goals
- Duration: max of 15 minutes
- Use the Scrum board
- Goal: Help team members coordinate their work, identify issues, and address dependencies
- Meet elsewhere to discuss management reporting or for problem-solving sessions

Iteration Review

- Demo your work to get feedback from PO and stakeholders
- Triggers: story done, spike done, refactor done, new NFR
- Format
    - State the iteration goals
    - Walk through the committed stories as a working, tested system
    - Spike: demo the findings
    - Reflect on stories not completed and why (uncovers impediments, risks, false assumptions, changing priorities, estimating issues, over-commitment)

Iteration Retrospective

- Led by Scrum Master
- Format
    - What went well?
    - What didn't go well?
    - What can we do better next time?

Building Quality In

Core to SAFe and Lean-Agile; reduce recall, rework, and defect fixing

Software Practices

- **Continuous integration (CI)**
    - Ideally merge to main several times a day to reduce risk of deferred integration
    - Minimum: local integration daily
    - Full system integration 1-2 times per interation
- **Test first**
    - Think about system behavior before implementing
    - Techniques: test-driven design (TDD) and behavior-driven design (BDD)
- **Refactoring**
    - Alter internal structure without changing external behavior
    - Essential to Agile development as the system changes over time
- **Pair/mob programming**
    - Multiple devs tackling issues (critical code, legacy code, interface definition, system-level integration)
    - Pair dev with tester
- **Collective ownership**

    - Anyone can add functionality, fix bugs, improve designs, or refactor
    - Don't rely on the original developer to be always available
    - One owner is a bottleneck (i.e., delay)

Firmware and Hardware Practices

- **Model-based systems engineering**
    - Use visual models to describe your system instead of lengthy documents
    - Focus: requirements, design, analysis, verification
- **Set-based design**
    - Think up many possible solutions
    - Continue weighing all possibilities until you gather enough data to narrow down your options
- **Frequent system integration** – find problems sooner
- **Design verification** – consider worst-case analysis of tolerances and performance, failure mode effects analysis (FMEA)

Improving Team Flow with Kanban

- Method for visualizing and managing work
- Aspects
    - Work moves through the system in a series of defined states
    - All work is visualized
    - Teams agree on WIP limits for each step; change limits to improve flow
    - Teams adopt policies (e.g., entry/exit criteria for a state, classes of service)
    - Flow is measured by lead time (request-to-completion) or cycle time (start-to-completion)
- **Cumulative flow diagram**
    - How much work is in what state over time
    - Vertical gap between spaces = how much WIP in that state
    - Horizontal gap between New and Done = lead time
- Classes of service
    - Standard – normal prioritizing
    - Fixed date – prioritize to meet the deadline
    - Expedite – high priority (ASAP)
- Kanban teams are on the train
    - When to use: uneven arrival, fast-changing priority, lower value of "what's getting done this sprint"
    - They still participate in ceremonies
    - They still have a role in rough estimation of work
    - Velocity can help determine throughput for larger planning, roadmapping

Ch 9: Executing the Program Increment

Overview

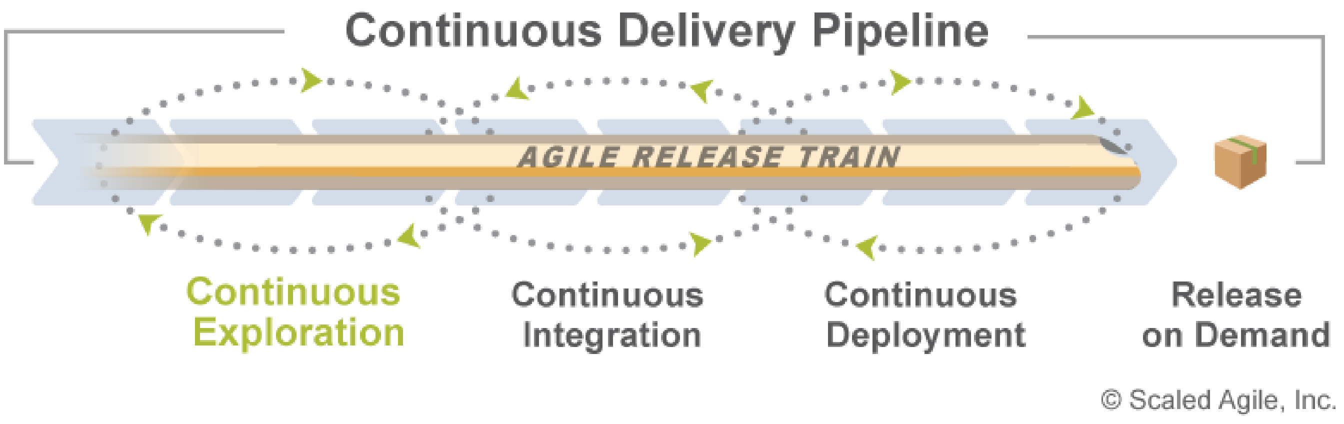
The **Agile Release Train (ART)** is key for delivering value.

- Made mostly of Agile teams
- Exists to build solutions that deliver business benefit
- Long-lived, runs continuously
- Cross-functional to continuously explore, execute, and release value over the PI

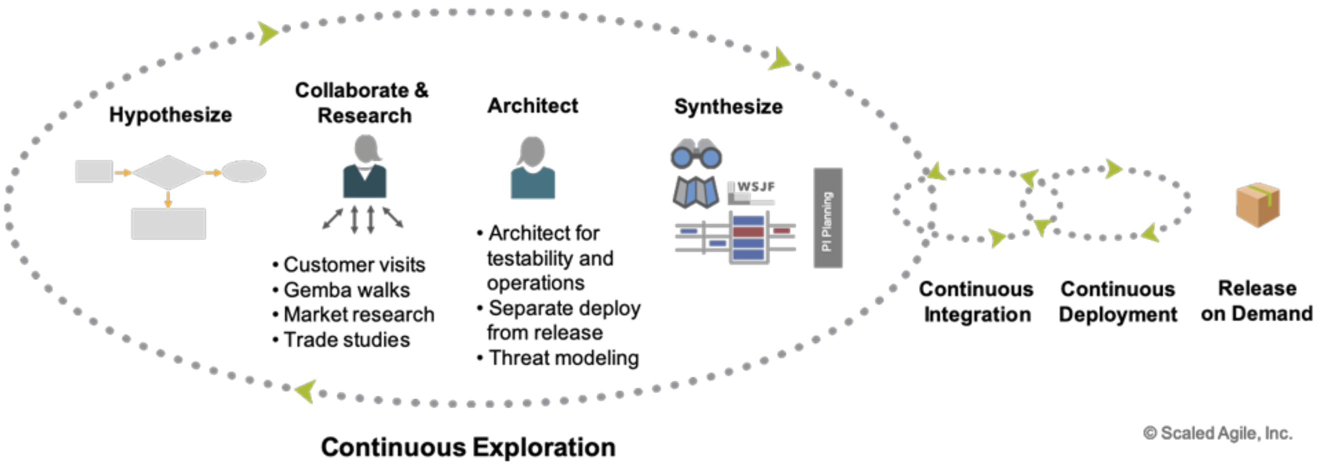
Note: The book covers SAFe 4.5, but at the time of these notes the SAFe website uses SAFe 5, so I've adapted the diagrams and descriptions from the website.

Continuous Delivery Pipeline

- Pipeline = workflows, activities, automation
- Goal: deliver small batches of new functionality on business demand
- Overlapping continuous activities: exploration, integration, deployment

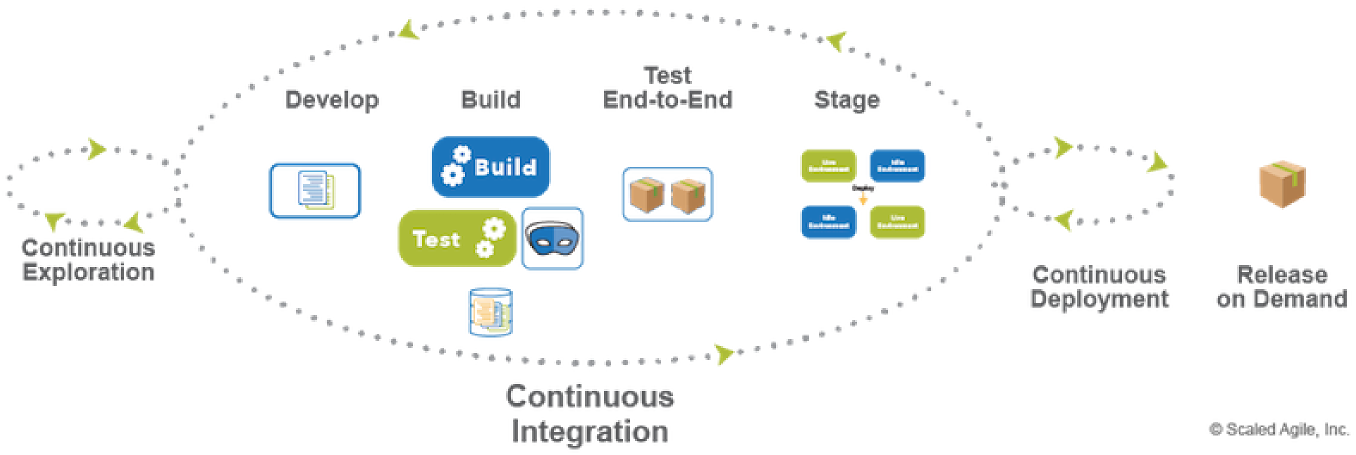


Continuous Exploration



1. **Hypothesize** – identify ideas, and the measurements needed to validate them with customers
2. **Collaborate & Research** – work with customers and stakeholders to refine the understandings of potential needs
3. **Architect** – envision a technological approach that enables quick implementation, delivery, and support of ongoing operations
4. **Synthesize** – organize the ideas into a holistic vision, a roadmap, a prioritized program backlog, and supports final alignment during PI Planning

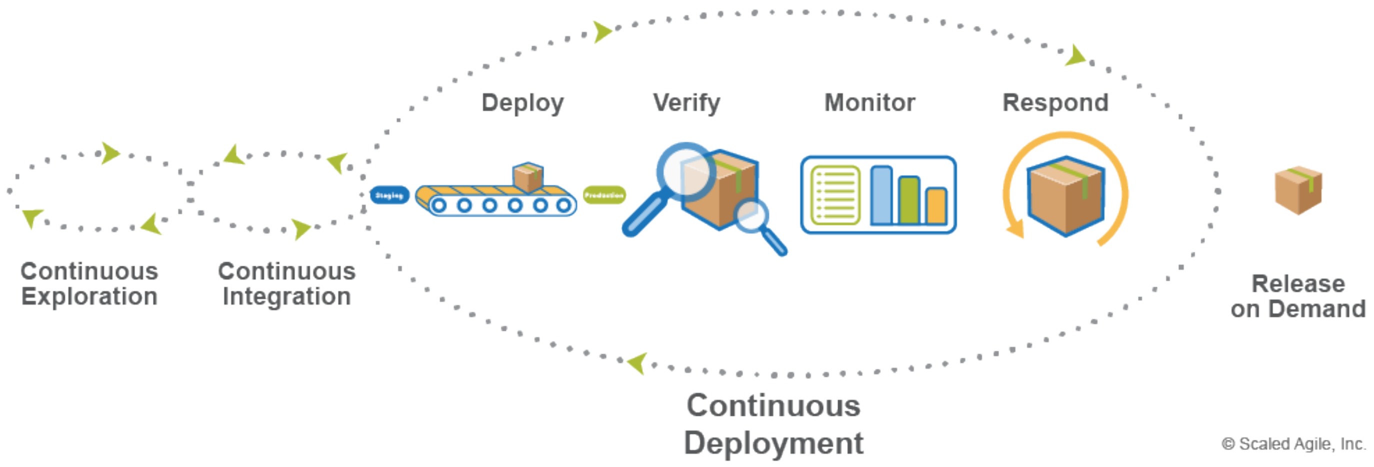
Continuous Integration



- **Develop** – implement stories and commit the code and components to the trunk
  - Common practices: features stories, BDD, TDD, version control, built-in quality, application telemetry, threat modeling
- **Build** – create deployable binaries and merge the development branches into the trunk

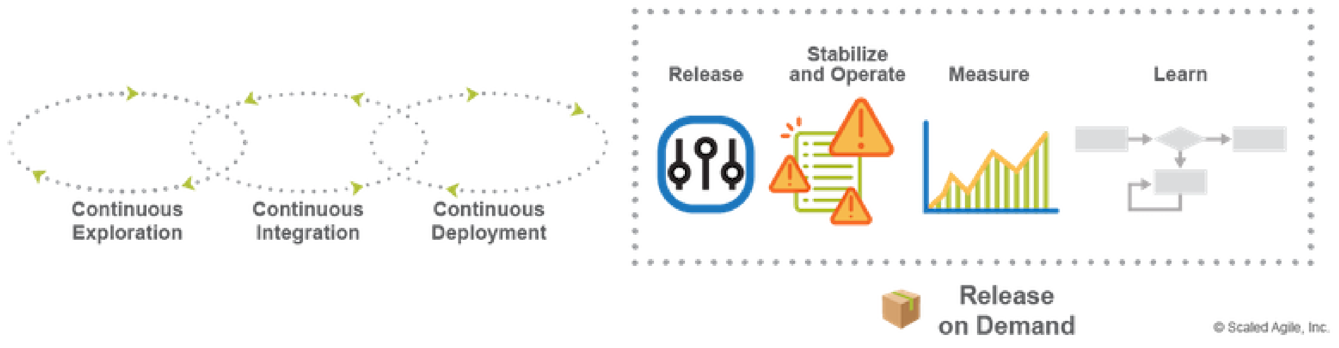
    - Common practices: commit often, gated commits (code compiles, tests pass before committing), avoid long-lived branches, automated tests, security inspection via code analyzers
- **Test end-to-end** – validate the solution
    - Common practices: test environments look like prod, automate as much testing as possible, relevant and stable test data, virtualized services to mimic production services, testings NFRs, continuous integration with suppliers
- **Stage** – host and validate the solution in a staging environment before production
    - Common practices: staging looks like prod, blue/green deploys, system demo to stakeholders

Continuous Deployment



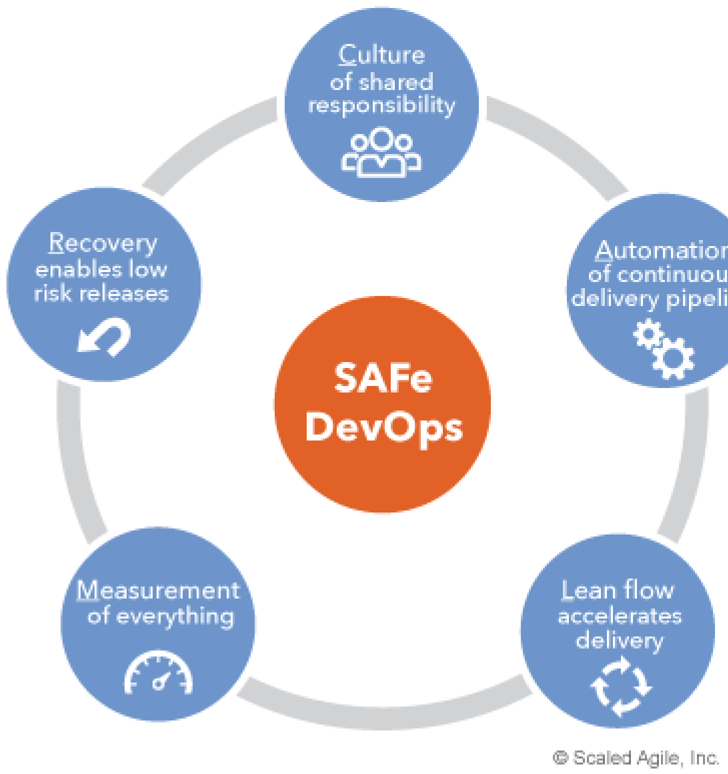
- **Deploy to production**
    - Common practices: dark launches, feature toggles, automated deploys, selective deployment (e.g., A/B, regional), ability to recover, blue/green deploys
- **Verify the solution** – make sure the changes operate in production as intended before they are released to customers
    - Common practices: production testing, automated testing, good test data, testing NFRs
- **Monitor for problems** – monitor and report on any issues that may arise in production
    - Common practices: full-stack telemetry, visual displays, federated monitoring (dashboards)
- **Respond and recover** – rapidly address any problems that happen during deployment
    - Common practices: proactive detection, cross-team collaboration, session replay, rollback and fix forward, immutable infrastructure (manage infra through pipeline), version controlled environments

Release on Demand



- **Release** – deliver the solution to end users, all at once or incrementally
    - Common practices: dark launch, feature toggle, canary release, release elements that are independent
- **Stabilize and operate** – make sure the solution is working well from a functional and non-functional perspective
    - Common practices: cross-team collaboration, failover/disaster recovery, continuous security monitoring, consider operational needs (e.g., high load), monitor NFRs
- **Measure** – quantify whether the newly-released functionality provides the intended value
    - Common practices: application telemetry, innovation accounting
- **Learn** – decide what should be done with the information gathered and prepare for the next loop through the continuous delivery pipeline
    - Common practices: Lean startup thinking, value stream mapping to find bottlenecks, relentless improvement

Enabling Continuous Delivery with DevOps


Culture of shared responsibility
Automation of continuous delivery pipeline
Lean flow accelerates delivery
Measurement of everything
Recovery enables low risk releases
SAFe DevOps
© Scaled Agile, Inc.

- **Culture of shared responsibility** – ops gets shifted upstream, dev gets shifted downstream; everyone helps deliver
- **Automation of the CD pipeline** – manual processes kill speed, productivity, and safety; you want things that are repeatable, self-documenting, more secure, more easily automated
- **Lean flow accelerates value delivery** – continuous flow of features to cash; visualize WIP, reduce batch sizes, manage queue lengths
- **Measurement of everything** – real-time telemetry to assess frequent changes
- **Recovery enables low-risk releases** – design for low-risk component/service-based deployability, releasability, and fast recovery; consider andon chord, roll back, plan for failures, chaos engineering

Enabling Continuous Delivery with Architectural Runway

- **Architectural runway** = having sufficient infrastructure to implement highest-priority features in near-term without excessive redesign and delay
- Enablers features/stories (tech debt, performance improvements) are prioritized to extend the runway

Managing Continuous Delivery with the Program Kanban

- Managed by Product
- Master backlog of work moving through the phases (idea funnel all the way to complete)

Supporting Continuous Delivery with Program Events

- This is about keeping the ART on the tracks
- **PI Planning**
- **Scrum of Scrums** (coordinate ART dependencies) or **PO Sync** (is ART progressing); sometimes called the **ART Sync**
- **System Demo** (usually done at Inspect and Adapt)
- **Prepare for PI Planning**
- **Inspect and Adapt** (retro of retros, what changes do we make, what things in the system need changing)
- **Innovation and Planning Iteration**
    - Comes at the end of the PI (like a sprint after the four delivery sprints)

  - Innovate and explore; not focused on delivery
  - Continuous learning
  - Includes PI planning, backlog refinement
  - If you’re using this time to complete previous PI work, you have something to fix in your process.

Ch 10: Inspect and Adapt

Overview

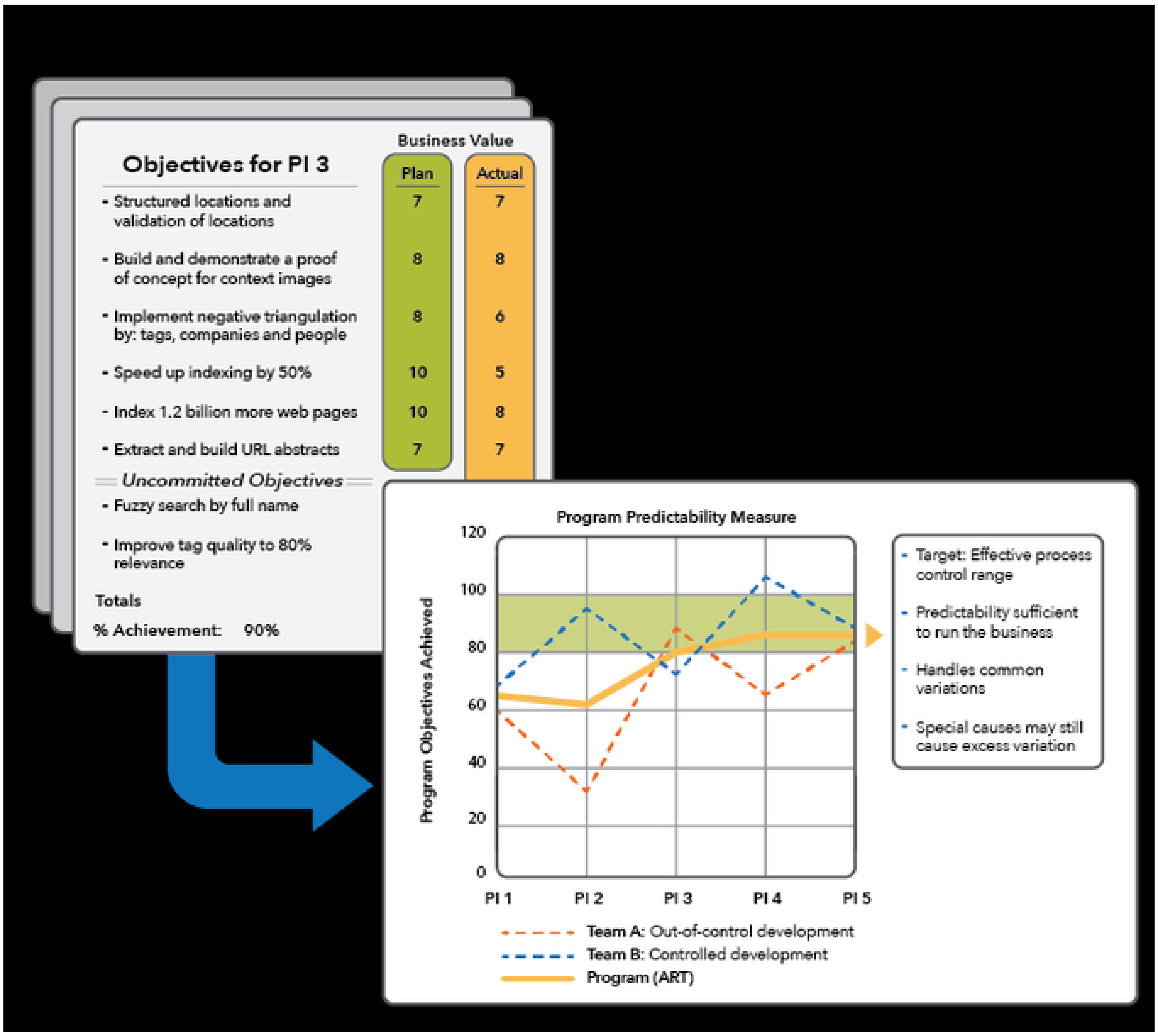
- Held after the end of each PI
- Demonstrate and evaluate the current state of the full solution
- Goal: Find improvement items so that every ART improves every PI
- Stakeholders: Agile team, RTE, System Architect, Product, Business Owners

Part 1: PI System Demo

- Show **all the features** done in the past PI
- More formal, requires more prep
- Time-box to 1 hour so you get feedback from your audience

Part 2: Quantitative Measurement

- Teams review metrics and trends; typically done by Scrum Masters and RTE
- This is a progress report of planned vs. actual value achieved
- SAFe prescribes that reliable trains generally operate between 80-100%



Objectives for PI 3

	Business Value Plan	Business Value Actual
- Structured locations and validation of locations	7	7
- Build and demonstrate a proof of concept for context images	8	8
- Implement negative triangulation by: tags, companies and people	8	6
- Speed up indexing by 50%	10	5
- Index 1.2 billion more web pages	10	8
- Extract and build URL abstracts	7	7
Uncommitted Objectives		
- Fuzzy search by full name		
- Improve tag quality to 80% relevance		
Totals		
% Achievement: 90%		

Program Predictability Measure
Program Objectives Achieved
120
100
80
60
40
20
0
PI 1
PI 2
PI 3
PI 4
PI 5
Team A: Out-of-control development
Team B: Controlled development
Program (ART)
- Target: Effective process control range
- Predictability sufficient to run the business
- Handles common variations
- Special causes may still cause excess variation

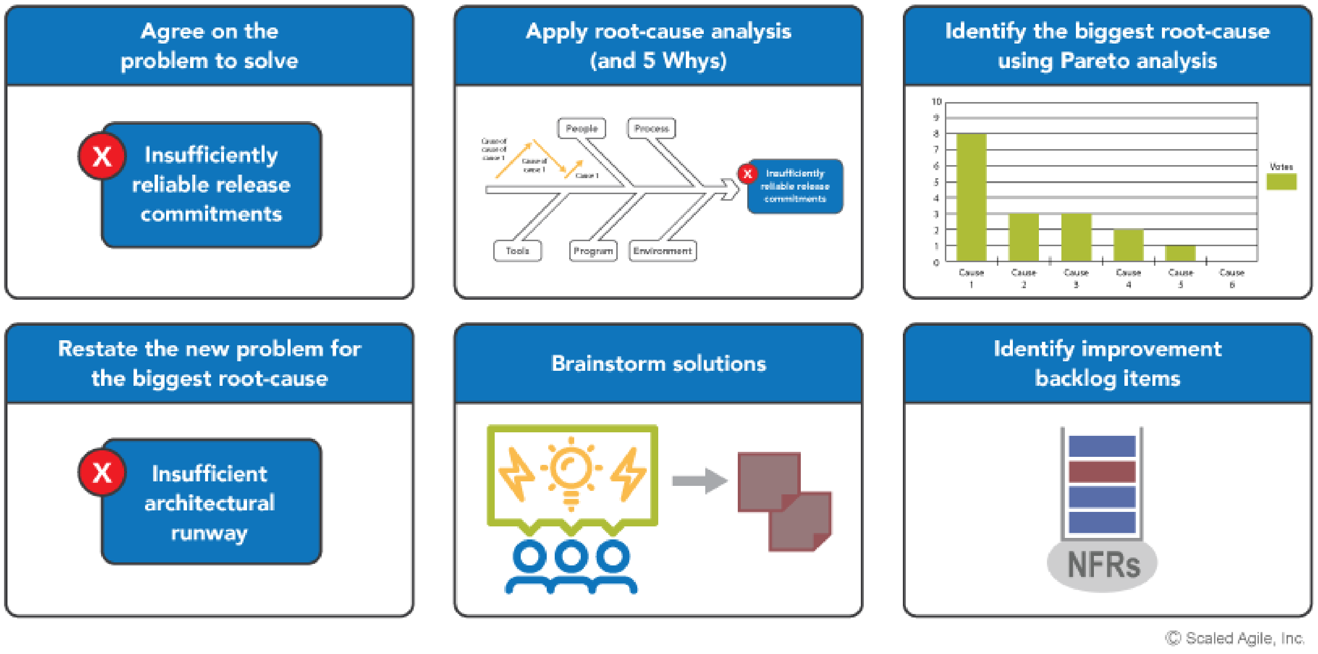
Part 3: Retrospective and Problem-Solving Workshop

Retrospective

- Identify broader program impediments to address
- Keep it limited to a few items to potentially address; usually 30 minutes
- Agree at a high level what to implement

Problem-Solving Workshop

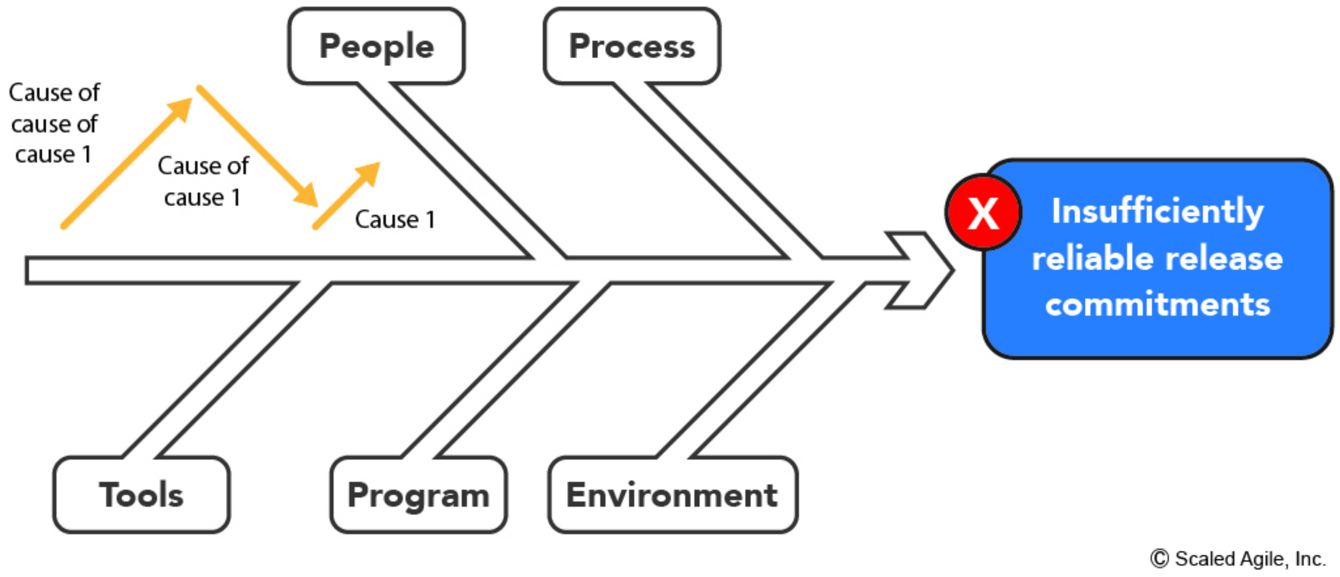
- Format: structured root-cause analysis to identify origins


Agree on the problem to solve
Insufficiently reliable release commitments
Apply root-cause analysis (and 5 Whys)
People
Process
Tools
Program
Environment
Identify the biggest root-cause using Pareto analysis
Votes
Cause 1
Cause 2
Cause 3
Cause 4
Cause 5
Cause 6
Restate the new problem for the biggest root-cause
Insufficient architectural runway
Brainstorm solutions
Identify improvement backlog items
NFRs
© Scaled Agile, Inc.

Agree on the Problems to Solve

- Put together a problem statement – what/when/where/impact
- Ex: "We discovered three significant design problems in the October deployment of the new EMV vehicles at the Thrills Amusement Park. The design flaws cause us to recall the vehicles and invest three months in materials, redesign, and testing. We delivered late, paid substantial penalties, and lost credibility with the customer."

Perform Root-Cause Analysis



Identify the Biggest Root Cause

- There are often many root causes
- **Pareto analysis** lets you identify which one is most common
- Let the team vote on which things are most relevant; summarize findings and assure consensus

Restate the New Problem

- …based on the root cause

Brainstorm Solutions

- Generate ideas, but don't debate yet
- Combine and mutate ideas

Create Improvement Backlog Items

- Vote on up to three potential solutions; these go into the Program Backlog

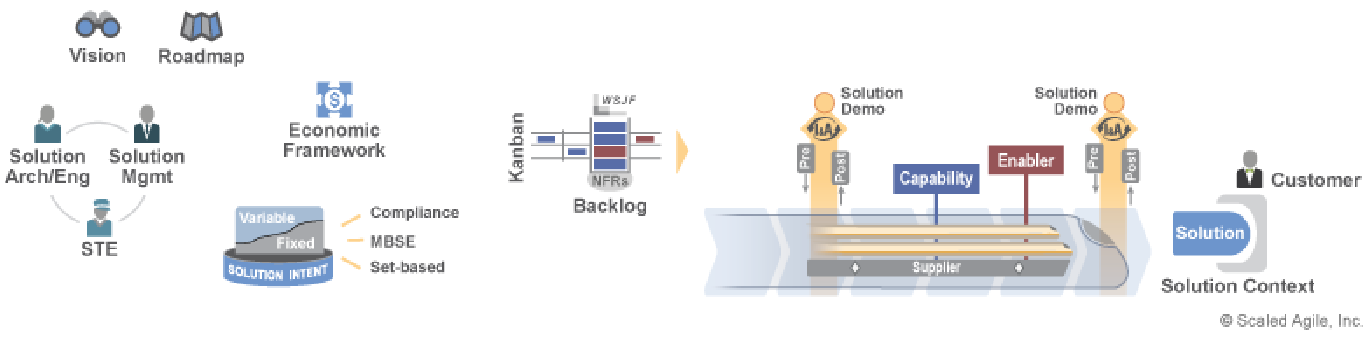
Ch 11: Large Solution SAFe Overview

Overview

- **Agile Release Train** = team of teams
- **Solution Train** = team of team-of-teams

The Solution Train

- Coordinates multiple ARTs and suppliers



- Focus on capturing requirements in solution intent
- Helpful for compliance, regulations, and standards; some companies may not need all this rigor, though
- Roles
  - **Solution Train Engineer (STE)** – servant leader and coach; supports and facilitates ARTs and suppliers
  - **Solution Management** – represents customer's overall needs; communicates strategic themes and portfolio vision
  - **Solution Architect/Engineer** – defines tech and architecture than connects the solution across trains

Solution Intent

- Usually at this scale the cost of failure is high; need more rigor around definitions and validation of system behavior
- **Solution intent** – single source of truth, repo for specs as they become clearer
  - Compliance – built-in quality, meets industry standards using Lean-Agile development
  - Model-based Systems Engineering (MBSE) – how emergent requirements and design are developed, documented, maintained
  - Set-based Design (SBD) – preserve options and defer decisions to the last responsible moment

Capabilities and the Solution Backlog

- **Capability** – higher-level behavior that spans multiple ARTs, often several suppliers
  - Contains a benefit hypothesis, acceptance criteria, fits within a single PI
  - Has associated enabler capabilities
  - Developed, analyzed, and approved using the solution Kanban
  - Prioritized with other capabilities based on WSJF
- Split into features, then user stories

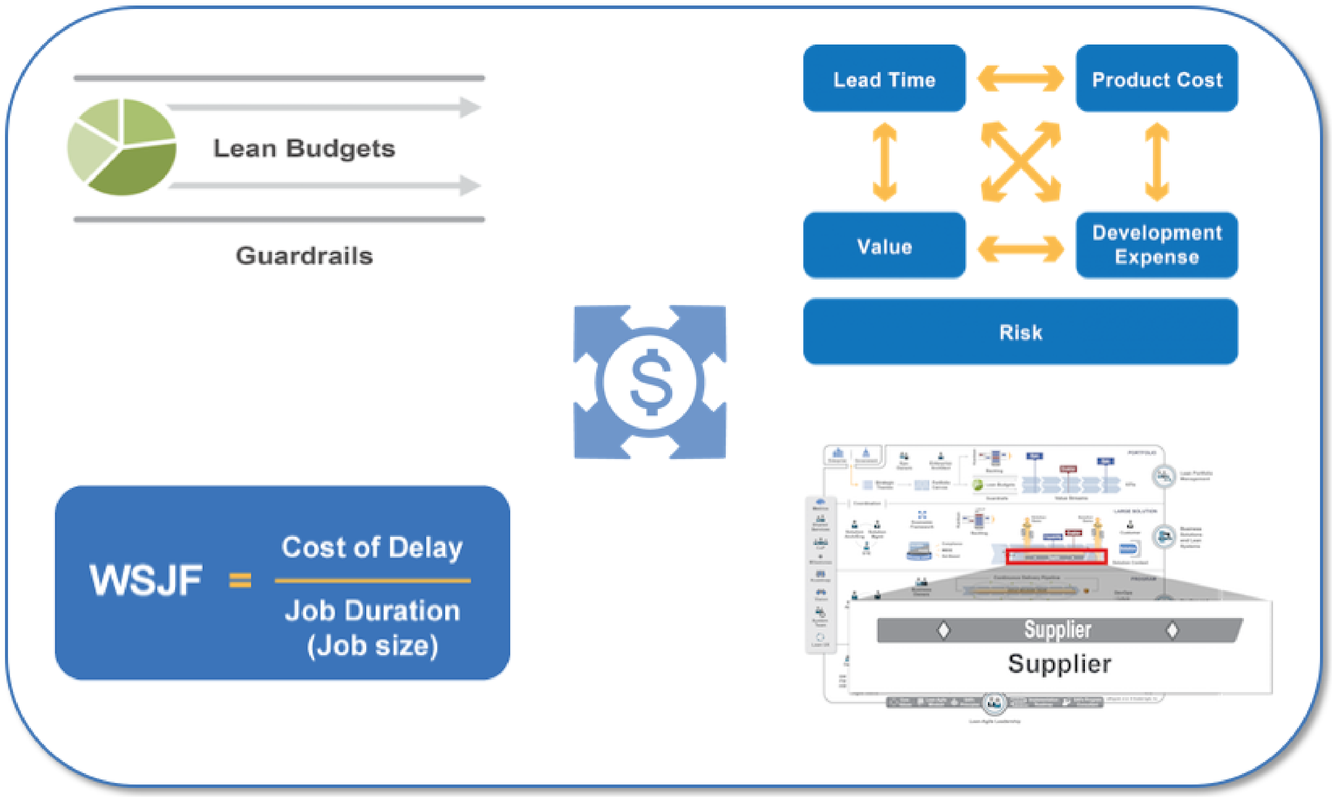
- *(Note: In Lirio's current work breakdown structure, I'm not sure where these would fall. The book goes on to describe epics which span PIs. Our initiatives can span epics.)*

Solution Epics

- Epics can span multiple PIs
- Can arise from splitting portfolio epics
- See Ch 14: Lean Portfolio Management

Economic Framework

Designed to permit fast, effective decision-making within the bounds of the broader economic requirements



© Scaled Agile, Inc.

- **Lean budgets** – moving from project-based, cost-center accounting to a process that deals with long-lived value streams; see Ch 15: Strategy and Investment Funding
- **Epic funding and governance** – empowered funding comes with the responsibility to communicate any investments that aren't routine
- **Decentralized economic decision-making** – collaborate broadly to synchronize, but you have Solution Management for Solution Trains, PMs for ARTs, and POs for teams
- **Job Sequences Based on Cost of Delay** – sequence things based on program and solution Kanban systems; pull using WSJF

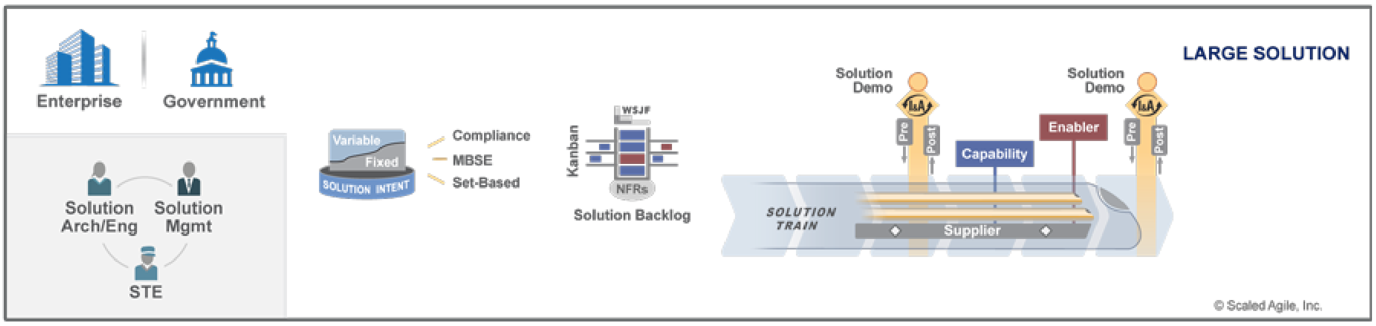
Ch 12: Defining Large and Complex Solutions

Overview

- High cost of failure = common barrier to Agile adoption (seems to conflict with "working software > comprehensive docs")
- Larger systems often need more records (e.g., trade studies, experiments, rationale for choices)
- To manage complexity and intensity…
  - **Solution** – products, systems, services
  - **Intent** – repository for solution knowledge
  - **Context** – ecosystem in which the solution operates

Solution

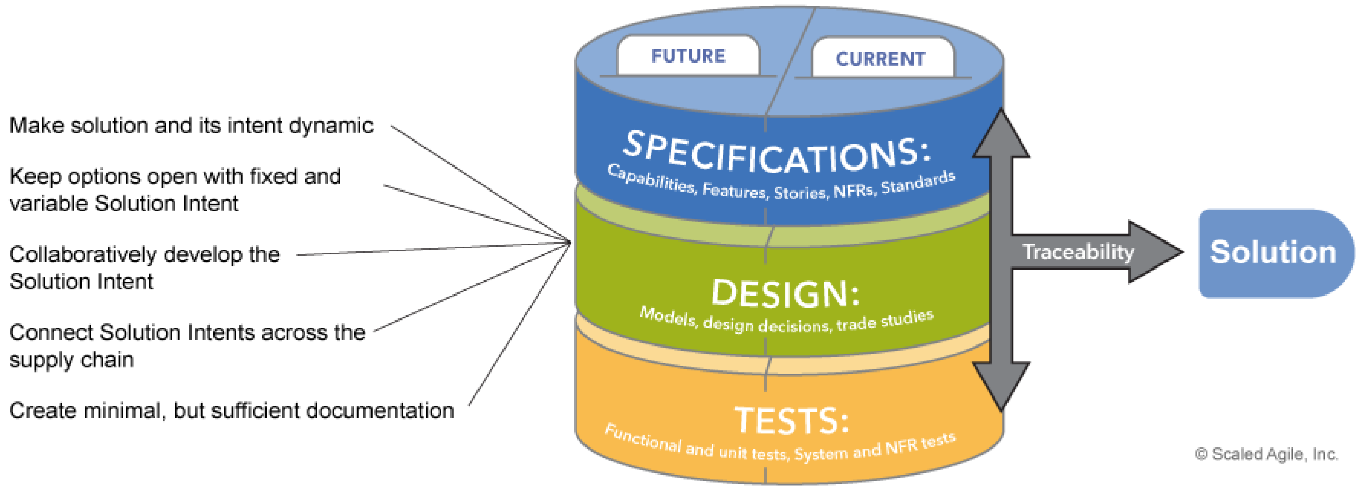
- Set of final products, systems, or services delivered to the external customer or enables the work of an internal value stream
- Requires multiple ARTs (i.e., Solution Train), multiple suppliers



- **Customers** interact with the dev team to clarify intent, validation assumptions review progress
- **Solution Management and Architects** help drive development, make scope and priority decisions, manage flow of features, capabilities, and NFRs in the Solution Kanban
- **Governance** comes from the economic framework from Ch 11: Large Solution SAFe Overview

Solution Intent

- Purposes
  - Provide a single source of truth for behavior
  - Record requirements, design, and architecture decisions
  - Facilitate exploration and analysis
  - Align customers, teams, and suppliers to a common understanding
  - Support compliance and contractual obligations



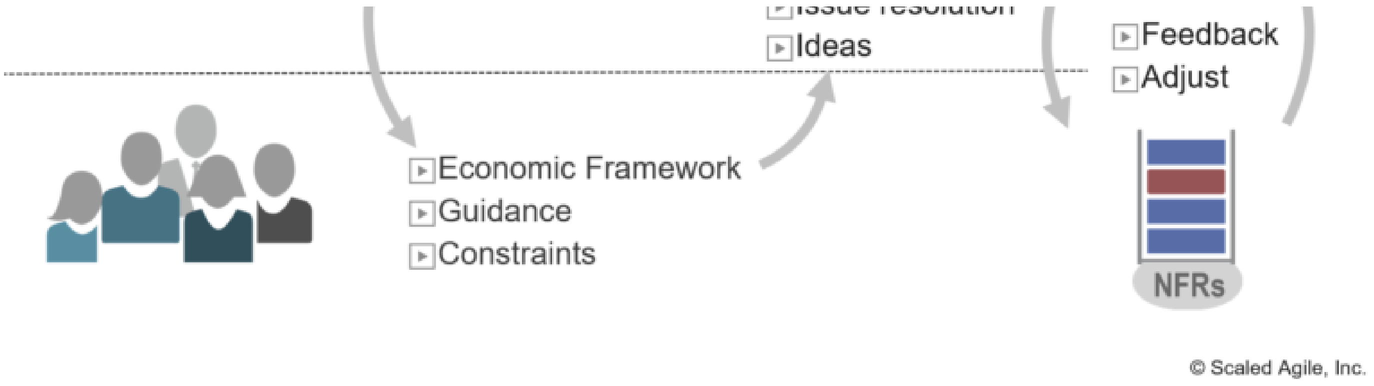
Make solution and its intent dynamic
Keep options open with fixed and variable Solution Intent
Collaboratively develop the Solution Intent
Connect Solution Intents across the supply chain
Create minimal, but sufficient documentation
FUTURE
CURRENT
SPECIFICATIONS:
Capabilities, Features, Stories, NFRs, Standards
DESIGN:
Models, design decisions, trade studies
TESTS:
Functional and unit tests, System and NFR tests
Traceability
Solution
© Scaled Agile, Inc.

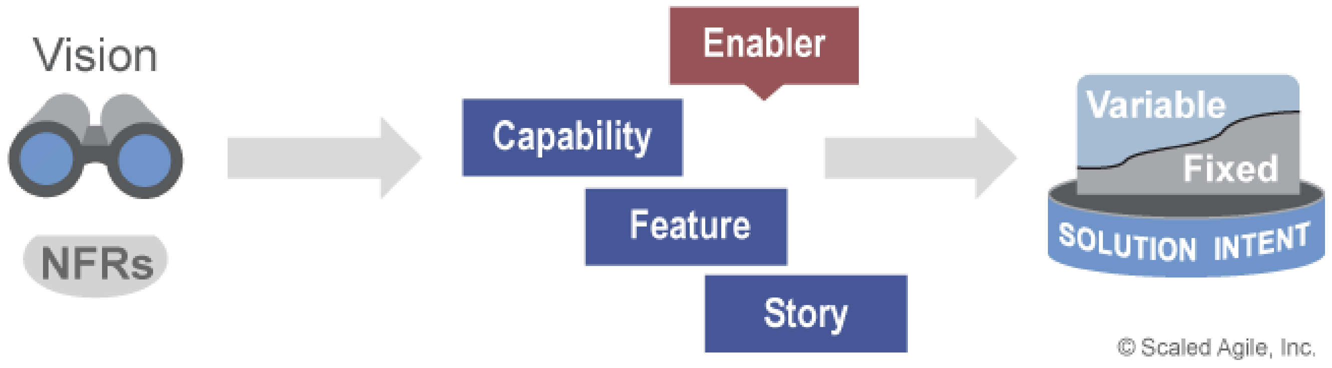
Fixed and Variable Solution Intent

- Recall principle #3 (assume variability, preserve options) – Ch 4: SAFe Principles
- **Fixed intent** – required or known behaviors
- **Variable intent** – explore tradeoffs and alternatives

Developing Solution Intent

Variable
Fixed
New knowledge
Key decisions
Issue resolution
Variable
Fixed
SOLUTION INTENT





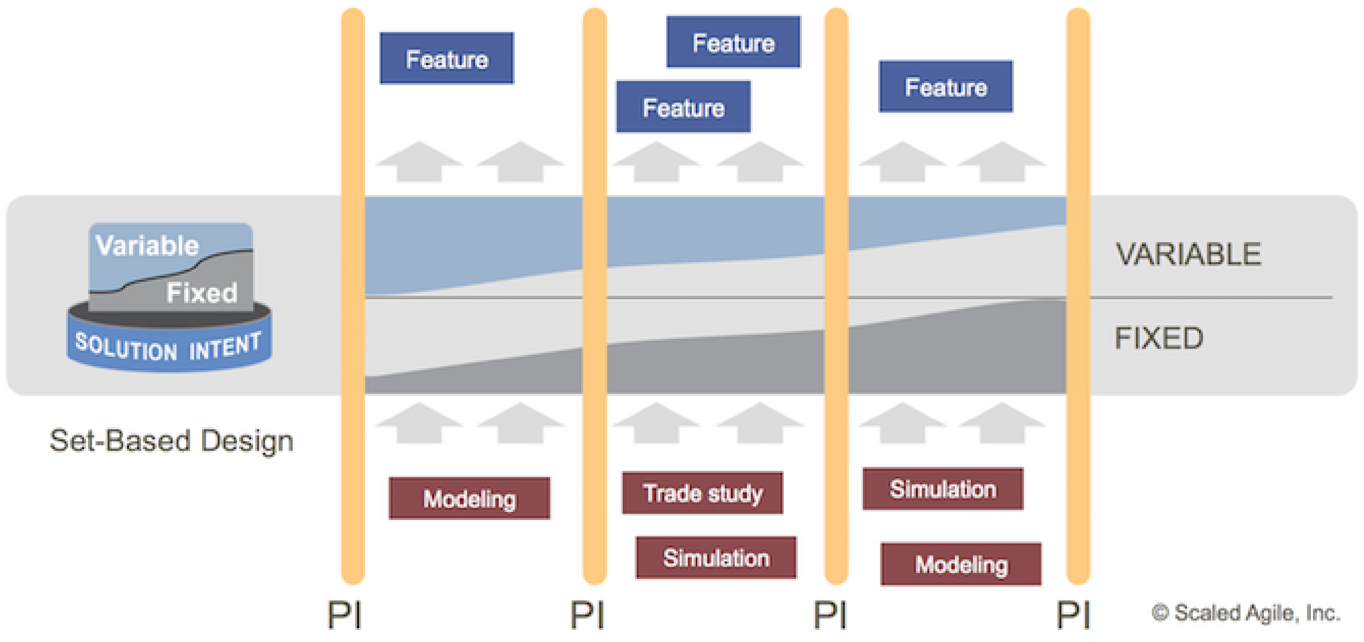
- Use MVPs to validate learning (leap/test/measure/pivot)

Collaborating on Solution Intent

- Solution Architect / Engineering – high-level system-wide decisions (decomposition, interfaces, subsystem capabilities)
- ARTs – take solution behaviors then influence program backlogs

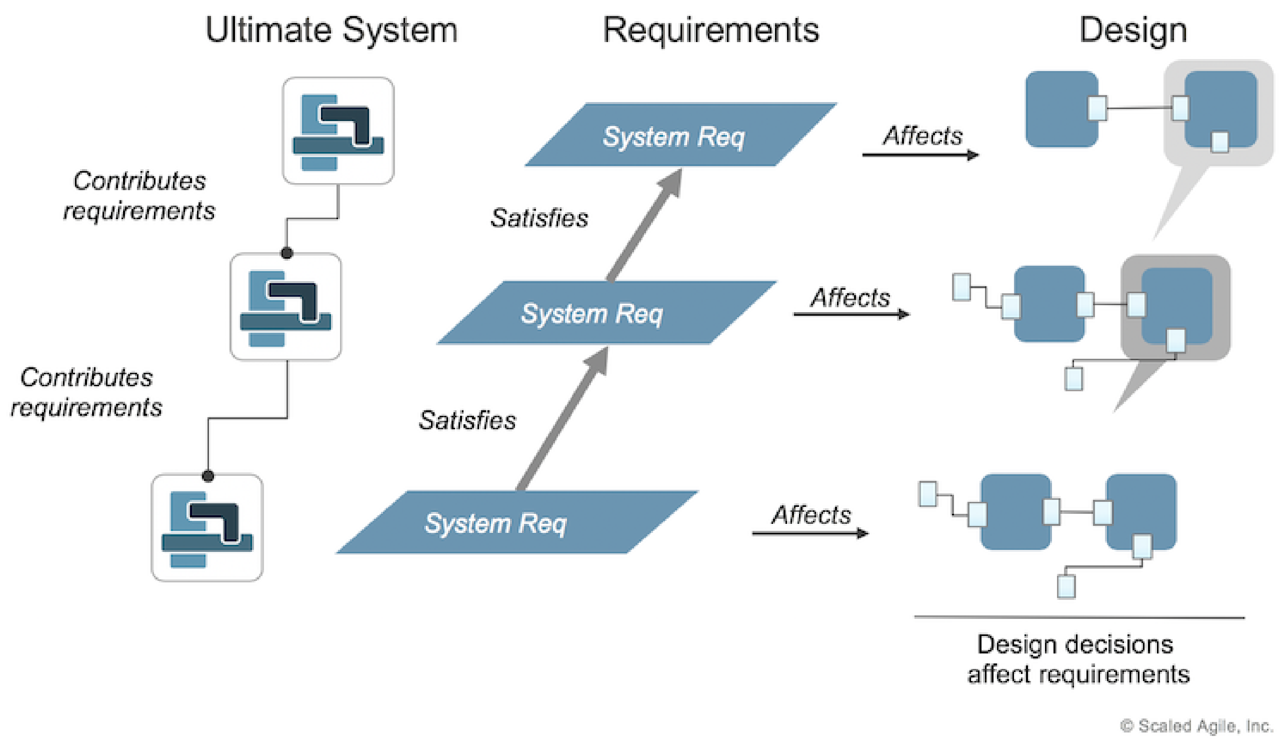
Moving from Variable to Fixed Solution Intent

- Unknowns must become knowns over time



System-of-Systems Solution Intent

- Intents don’t exist in a vacuum (e.g., suppliers have other customers as well, so if their design changes, their customers are impacted)



Minimum Responsible Documentation

- Models > Documents
- Keep solution intent collaborative
- Keep options open (defer to teams doing the work)
- Document in one place
- Keep it high-level
- Keep it simple; record only what’s needed

Solution Intent Documentation in High-Assurance Environments

- Regulated environments may require standards or other technical specs; some have requirements around traceability, decision decisions, etc.
- As long as these exist when they need to (i.e., instead of being all done up front), you can still be Agile yet compliant

Solution Context

- Identifies critical aspects of the target solution environment and its impact on usage, installation, operations, support, and even marketing, packaging, and selling
- Customer participates pre- and post-PI planning meetings and solution demos; validate assumptions as often as possible

Solution Context for a System of Systems

- Each part of the supply chain delivers its solutions to the customer’s context
- Example: navigation supplier info-tainment supplier vehicle manufacturer customer

Solution Context for IT Deployment Environments

- Internal customers still require context (interfaces, deployed OSes, firewalls, APIs, cloud infrastructure, etc.)

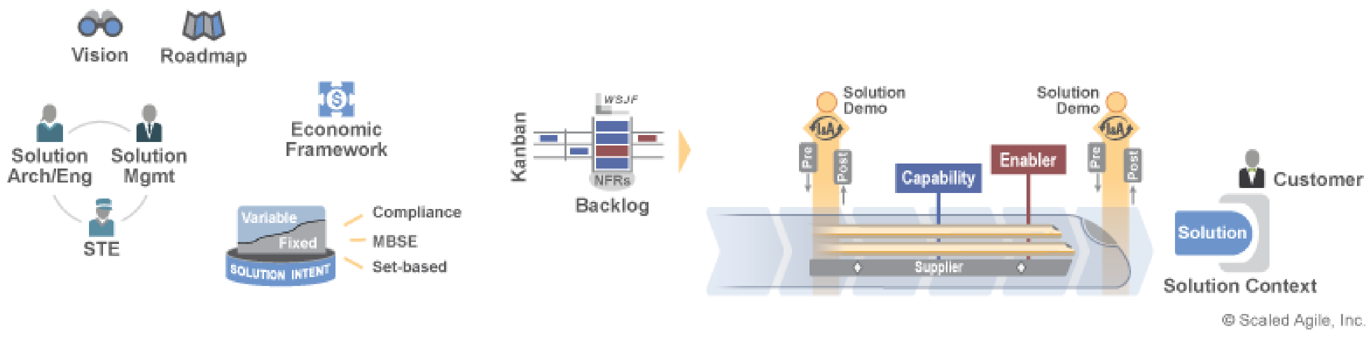
Solution Context Includes Portfolio-level Concerns

- Products and services of a business must work together to accomplish the org's broader objectives
- Solutions are part of a portfolio

Ch 13: Solution Train Execution

Overview

For really big systems, a single ART can't do the job. This is what Solution Trains are for.



Pre-PI Planning

- Establish broader solution context and goals for ART PI planning meetings
- Inputs
  - Solution vision & roadmap
  - Context from most recent solution demo
  - Updates to solution intent
  - Highest priority capabilities from the backlog
- Attendees
  - Solution Train Engineer (STE)
  - Solution Management
  - Solution Architect
  - Reps from other ARTs and suppliers
- Goal: Build context to create ART and supplier plans so individual PI planning events are successful
  - What's been achieved so far?
  - Executives match current solution to next desired state, milestones, etc.
  - Review top capabilities in the backlog
  - Sketch out next features, dependencies, and potential impacts on the solution

ART PI Planning

This is pretty much the same as Ch 7: Planning a Program Increment, but with the pre-PI planning output as a feeder to this step.

Post PI Planning

Now that the individual ARTs have done some planning, you get everyone back together.

- Read-out of individual ARTs, objectives, milestones
- Review the plan, talk about risk, and get a confidence vote
- Rework plans if needed
- Retrospective on how planning went
- Outputs
  - Set of PI objectives
  - Solution planning board

Frequent Solution Integration

- ARTs typically integrate every iteration, but this may be harder with multiple ARTs (multidisciplinary, large, and complex systems)
- The more frequent the better, as the longer you wait the more difficult it will be

Solution Train Sync

- This is basically Scrum of Scrums; weekly, 60 minutes

Solution Demo

- Scheduling is flexible because integration is usually harder; try not to skip this

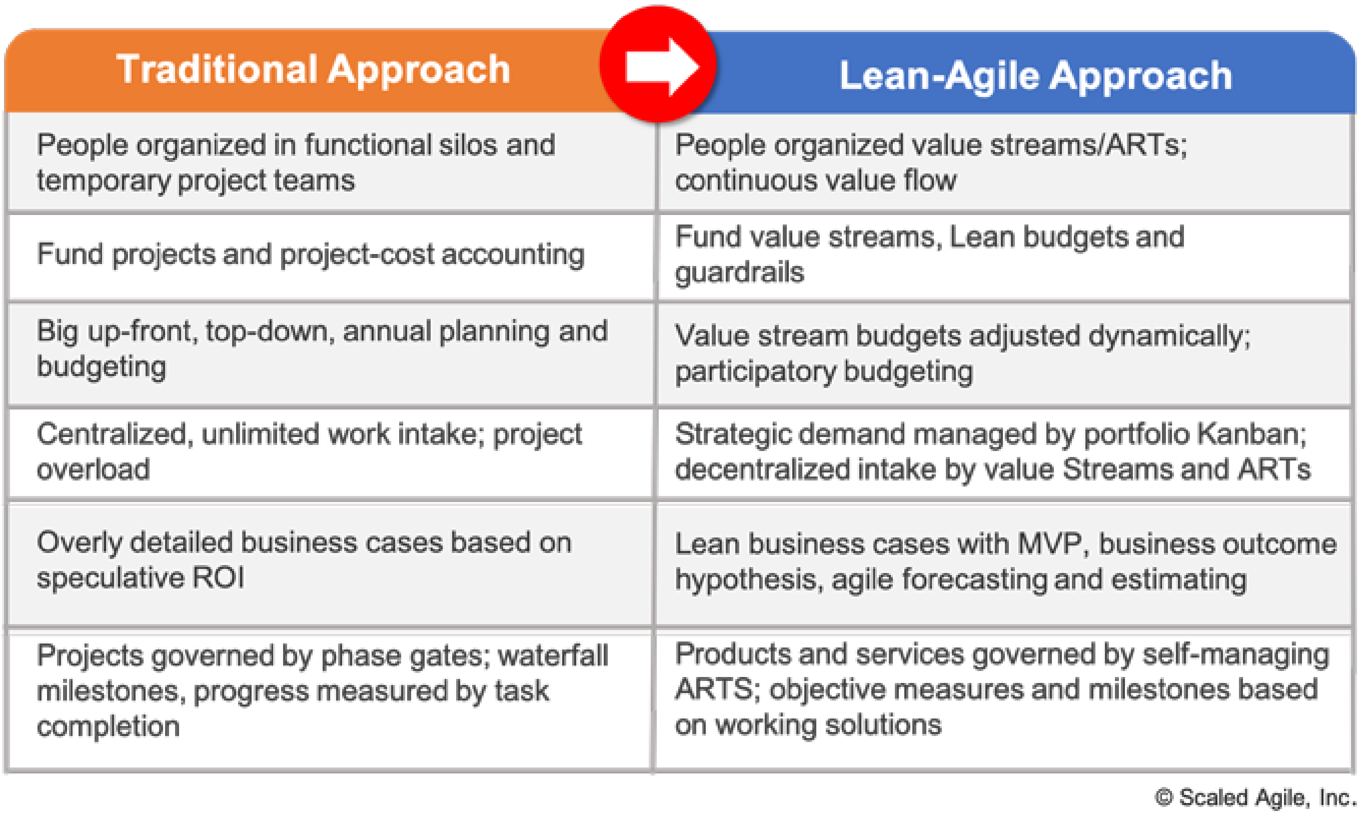
Solution Train Inspect and Adapt

- After every PI, do and I&A workshop (see Ch 10: Inspect and Adapt)
- Usually can't include everyone because solution trains are so large; get the main stakeholders and the RTEs
- Take the learnings and put them in the backlog

Ch 14: Lean Portfolio Management

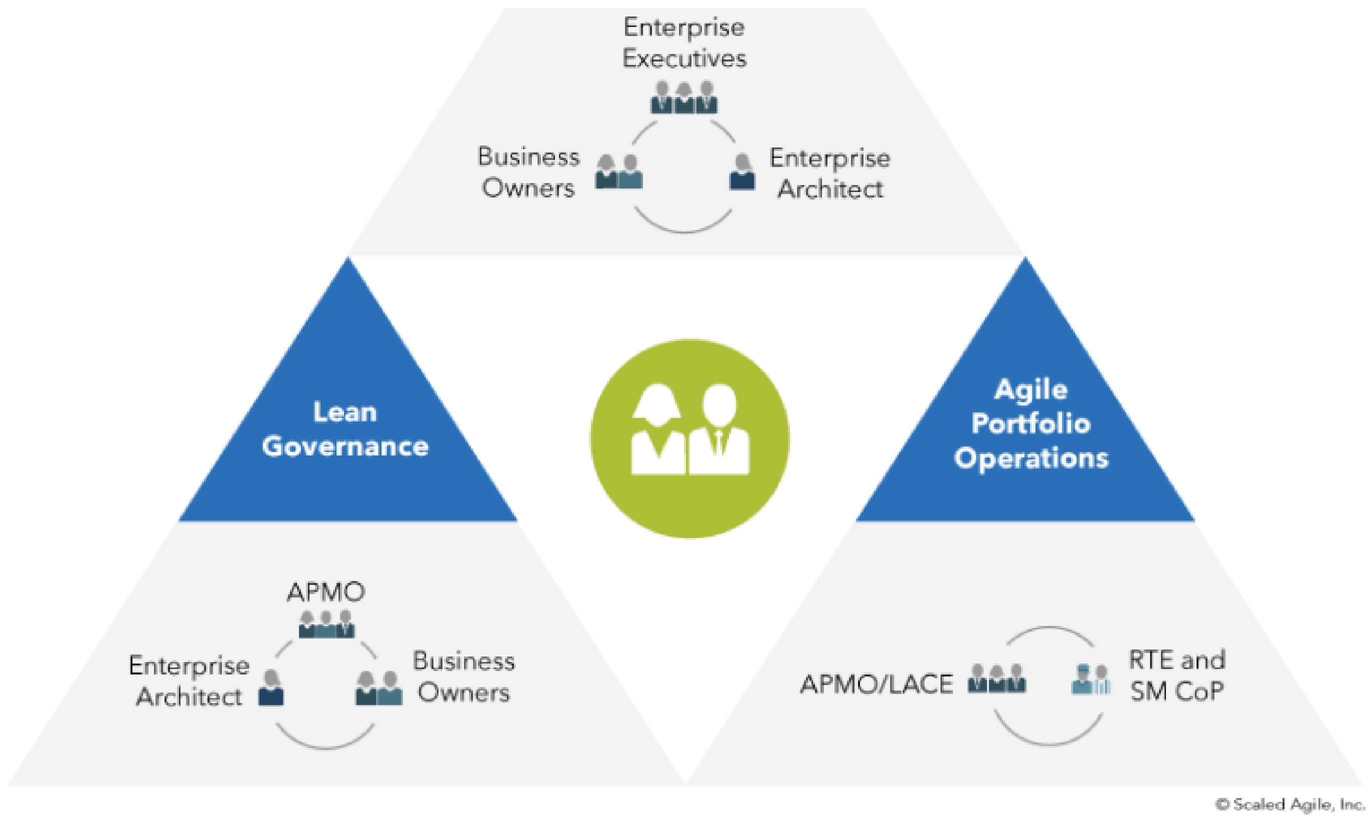
Introduction

Lean Portfolio Management (LPM) – highest level of decision-making and financial accountability for products and solutions in the SAFe portfolio



Traditional Approach	Lean-Agile Approach
People organized in functional silos and temporary project teams	People organized value streams/ARTs; continuous value flow
Fund projects and project-cost accounting	Fund value streams, Lean budgets and guardrails
Big up-front, top-down, annual planning and budgeting	Value stream budgets adjusted dynamically; participatory budgeting
Centralized, unlimited work intake; project overload	Strategic demand managed by portfolio Kanban; decentralized intake by value Streams and ARTs
Overly detailed business cases based on speculative ROI	Lean business cases with MVP, business outcome hypothesis, agile forecasting and estimating
Projects governed by phase gates; waterfall milestones, progress measured by task completion	Products and services governed by self-managing ARTS; objective measures and milestones based on working solutions

© Scaled Agile, Inc.



Strategy and Investment Funding

- Allocate investments to building the right things
- See Ch 15: Strategy and Investment Funding

Responsibilities (per SAFe 5)

- **Connect the portfolio strategy to enterprise strategy** – the enterprise needs the portfolio and vice versa
- **Maintain a portfolio vision**
  - Portfolio vision – future state of value streams and solutions; review quarterly (not one-and-done)
  - Enterprise architecture – effective technology plans (e.g., tech stacks, interoperability, APIs, hosting, security)
  - Portfolio roadmap – integrate lower-level (PI) roadmaps into a more comprehensive view; warning: "every long-term commitment decreases the agility of the organization"
- **Realize portfolio vision through epics**
  - Epics are more nebulous, so use Portfolio Kanban
  - Mix business epics and enabler epics
- **Establish lean budgets and guardrails**
  - Fund value streams, not projects
  - Guardrails support budgets by providing governance and spending policies/practices
- **Establish portfolio flow**
  - Big initiatives require collaboration between multiple value streams or ARTs
  - Limit the number of cross-cutting initiatives, limit WIP, reduce batch sizes, control queue lengths for long-term development items, monitor capacity

Agile Portfolio Operations

- See Ch 16: Agile Portfolio Operations
- Move power from the PMO to the ARTs and Solution Trains

Responsibilities

- Coordinate value streams – manage dependencies

- Support program execution – typically through the Agile Program Management Office (APMO), CoPs around RTEs and STEs, Scrum Masters
- Foster operational excellence – Lean-Agile Center of Excellence (LACE) may be standalone or part of the APMO; this link has a more thorough list of responsibilities

Lean Governance

- Manages spending, audit, compliance, expense forecasting, and measurement

Responsibilities

- Forecast and budget dynamically – replace long-range budget cycles; based on cadence as part of the Strategic Portfolio Review or Participatory Budgeting Events
- Measure portfolio performance – SAFe has several recommended metrics
- Coordinate continuous compliance – measure continuously instead of yearly or at the end of projects; SAFe also has guidance about regulatory/industry standards

Ch 15: Strategy and Investment Funding

Introduction

- This content overlaps with the section by the same name in Ch 14: Lean Portfolio Management, so I won't repeat that here.

Issues with Project-Based Accounting

- Drives overhead, creates temporary work for temporary people (i.e., people go back to their department silos when done)
- At odds with long-lived Agile teams and persistent knowledge acquisition
- Admin overhead
- Defensive stances about cost overruns for unforeseen technical challenges
- Constant personnel reassignments
- Projects require multiple budgets to build a single budget – slow and complex, ute-based planning, low program throughput, moves people to the work
- Typically works best when you identify all the work upfront (when the least amount of knowledge exists)
- Less agility; what happens if a project half-way through doesn't pan out?
- Innovation cannot happen without risk, technical uncertainty; guards around this (e.g., change control boards) add further delays
- Depending on how toxic the environment, things can get political, bonuses are at risk, the numbers get gamed to protect yourself or blame others

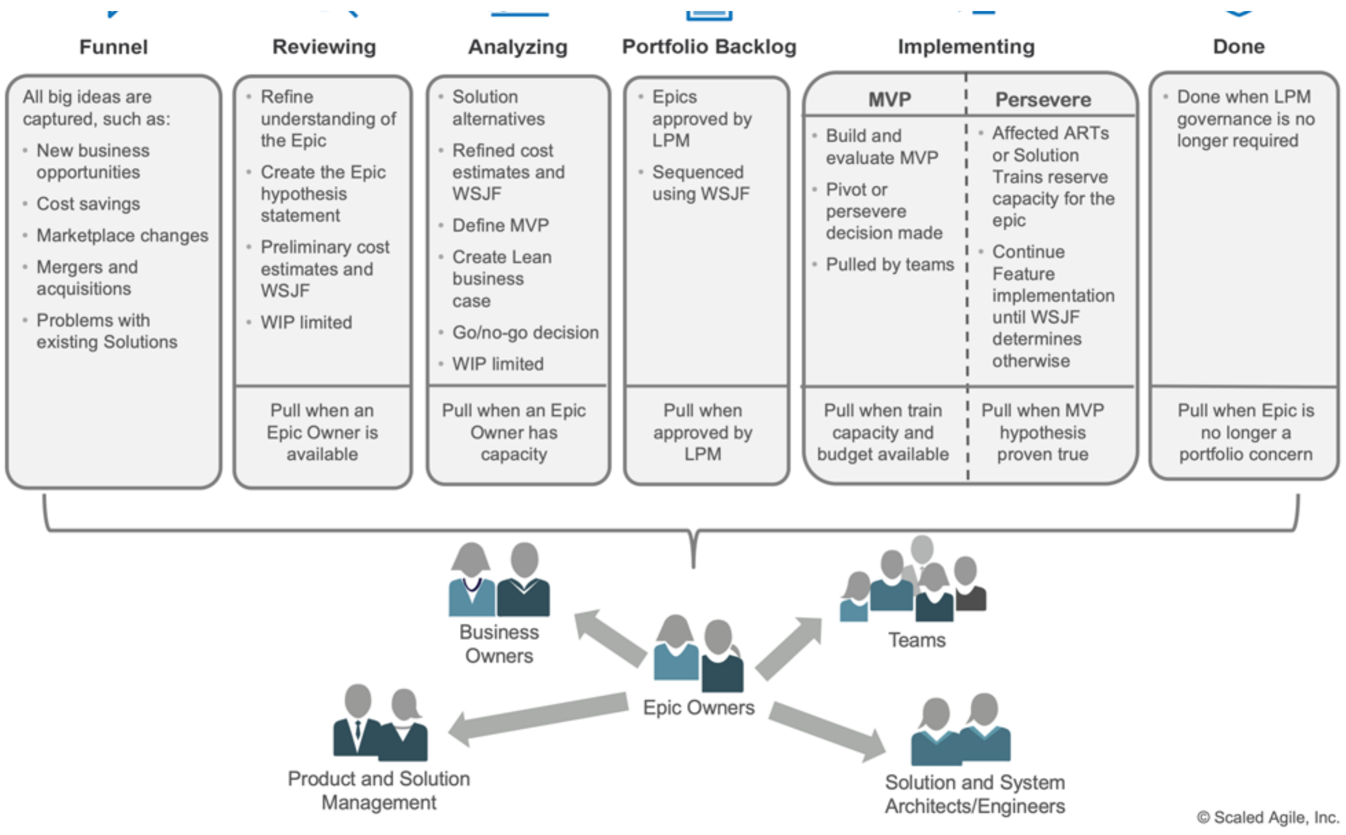
Fund Value Streams

- Use a fixed cost per PI
- If you plan two features, but only one gets done (with a few more bells and whistles) because it's the right decision for the business, you've done well.
- Benefits
    - Local empowerment – current backlog and roadmap are where the work lives
    - Higher throughput and improved morale – because of longer-lived teams
    - Full control of total spend – PI cost is fixed
    - Flexibility and agility – trains can flex to the work, focus on where the most value is

Lean Startup Cycle

- Hypothesize – lean business case about what the epic will deliver and what value is derived
- Build MVP – just enough to reject (or fail to reject) the hypothesis
- Evaluate MVP – what are the indicators of success
- Pivot or persevere – stop or add new features

Portfolio Kanban



Funnel
Reviewing
Analyzing
Portfolio Backlog
Implementing
Done
All big ideas are captured, such as:
New business opportunities
Cost savings
Marketplace changes
Mergers and acquisitions
Problems with existing Solutions
Refine understanding of the Epic
Create the Epic hypothesis statement
Preliminary cost estimates and WSJF
WIP limited
Pull when an Epic Owner is available
Solution alternatives
Refined cost estimates and WSJF
Define MVP
Create Lean business case
Go/no-go decision
WIP limited
Pull when an Epic Owner has capacity
Epics approved by LPM
Sequenced using WSJF
Pull when approved by LPM
MVP
Persevere
Build and evaluate MVP
Pivot or persevere decision made
Pulled by teams
Affected ARTs or Solution Trains reserve capacity for the epic
Continue Feature implementation until WSJF determines otherwise
Pull when train capacity and budget available
Pull when MVP hypothesis proven true
Done when LPM governance is no longer required
Pull when Epic is no longer a portfolio concern
Business Owners
Teams
Epic Owners
Product and Solution Management
Solution and System Architects/Engineers
© Scaled Agile, Inc.

- Business owners – synchronize priorities
- Product and solution management – split epics, prioritize features
- RTE – provide guidance for planning and execution by ARTs
- Agile Teams – coordinate research activities, help with implementation
- Architect – own enabler epics, help with runway, influence best practices and designs

Ch 16: Agile Portfolio Operations

Cast of Characters

Agile Program Management Office (APMO)

- Once things get large enough, moving all the work down to the ARTs and Solution Trains requires more coordination
- Focuses
  - Sponsors and communicates the change vision
  - Participates in the rollout
  - Leads the move to objective milestones
  - Helps implement Lean budgets
  - Fosters Agile contracts and learner supplier/customer partnerships
  - Provides support for effect program execution

Lean-Agile Center of Excellence (LACE)

- May be part of the APMO or a separate entity
- Focuses
  - Communicates the business needs urgency, and vision for change
  - Develops the implementation plan; manages transformation backlog
  - Establishes metrics and how they're communicated
  - Conducts coaching/training for execs, managers, leaders, teams
  - Identifies values streams and launches ARTs
  - Extends Lean-Agile practices to other areas of the company (budgets, portfolios, contracts, HR)
  - Establishes relentless improvement

- (This seems very much like the Software Engineering Process Group from CMMI.)
- Even though it doesn't have individual contributors per se, it operates like an Agile team – has a PO and scrum master, follows cadences
- Depending on the size, you can arrange things differently
  - Centralized – single portfolio, value streams and ARTs under a single budget
  - Decentralized – local LACEs, independent business units with autonomous SAFe portfolios, cross-business unit collaboration
  - Hub and spoke – large enterprise, core practices developed and shared, mix of central and local funding

RTE and Scrum Master CoPs

- Communities of Practice (CoPs) – organized groups of people with a common interest that collaborate regularly to share, improve, work on things
- (This is what Spotify calls *guilds,* or *chapters* if slicing by departments.)

Coordinate Value Streams

- Don't build the same thing twice or confuse the market
- Look for reuse of tech and tools
- Coordinate and facilitate availability of scarce skillsets or shared services (e.g., security, compliance)
- Sharing a cadence allows more regular synchronization

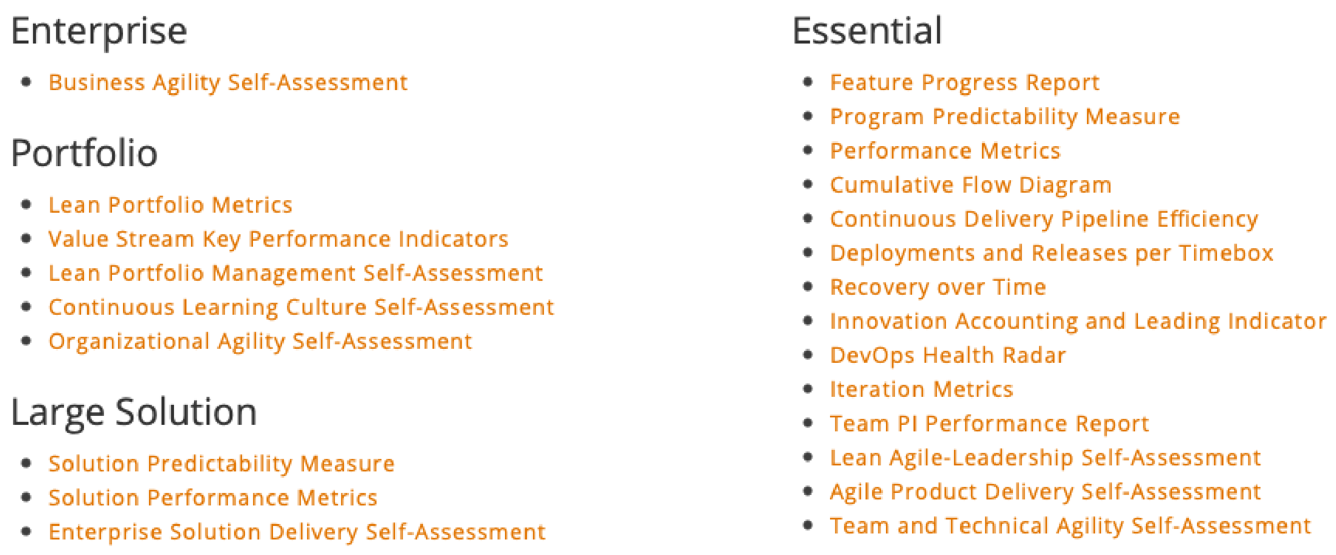
Ch 17: Lean Governance

Forecast and Budget Dynamically

- Just because Agile focuses on near-term delivery doesn't mean that forecasting has no value
- Estimation
  - According to the book (SAFe v4), there's a huge assumption that epics *are of known size* (they use story points as an example, but I've seen other companies use hour-ranges). They claim historical data is how you figure this out. Once the above assumption is made, you look at the velocities of your ARTs and their capacities. At this point, it's simple math to figure out how full those buckets can be. SAFe v5 doesn't seem to offer anything different.
  - **Dynamic budgeting** is about ratios of certain types of work (value streams) that may shift PI to PI. Lirio does this retrospectively by looking at value areas by story point for a given sprint to ensure we're not lop-sided in one particular area (e.g., compliance).

Measure Portfolio Performance

- SAFe has a vast set of metrics that can be used to measure performance



Enterprise

- Business Agility Self-Assessment

Portfolio

- Lean Portfolio Metrics
- Value Stream Key Performance Indicators
- Lean Portfolio Management Self-Assessment
- Continuous Learning Culture Self-Assessment
- Organizational Agility Self-Assessment

Large Solution

- Solution Predictability Measure
- Solution Performance Metrics
- Enterprise Solution Delivery Self-Assessment

Essential

- Feature Progress Report
- Program Predictability Measure
- Performance Metrics
- Cumulative Flow Diagram
- Continuous Delivery Pipeline Efficiency
- Deployments and Releases per Timebox
- Recovery over Time
- Innovation Accounting and Leading Indicator
- DevOps Health Radar
- Iteration Metrics
- Team PI Performance Report
- Lean Agile-Leadership Self-Assessment
- Agile Product Delivery Self-Assessment
- Team and Technical Agility Self-Assessment

- Innovation accounting – "quantifies the market value of new business opportunities that are fundamentally ambiguous and uncertain – the breakthroughs and disruptors."

Capitalization of Agile Software Development

- Many US companies are subject to US FASB (Financial Accounting Standards Board) regulations; capitalize software dev costs when a project/product meets certain criteria. (See your accountant for details.)
- Considerably more detail can be found on the SAFe site here

Governance via Agile Contracts

- Large systems are rarely built in-house, there are various suppliers
- Common approaches
  - Firm fixed price (most risk to the supplier)
  - Target price
  - Cost plus
  - Time and materials (most risk to the customer)
- Phase 1: Pre-commitment
  - Customer: understand constructs and responsibilities, defines program mission to the supplier
  - Supplier: analyzes potential feasibility, assures customer that they can deliver on the needs with a rough cost estimate
  - Shared: establish roadmap and vision, define fixed solution and variable solution, establish economic framework, establish responsibilities and contract boundaries, prioritize PI 1 backlog, determine MVP
- Phase 2: Execution
  - PI prep
  - PI planning
  - PI execution
  - PI evaluation with I&A event
- Both parties trust and verify that they are on the path to the best economic outcomes (long-term benefit to both)
- More info from SAFe about Agile contracts

Coordinate Continuous Compliance

- Most companies have relied on comprehensive quality management systems (QMS) based on phase-gated development models to reduce risk and ensure compliance (e.g., ISO 9001, CMMI)
- Compliance activities are typically deferred until the end of the project
- Lean QMS principles
  - Build the solution of compliance incrementally
  - Organize for value and compliance
  - Build in quality and compliance
  - Continuously verify and validate
  - Release validated solutions on demand

Ch 18: The Guiding Coalition

Introduction

- From John Kotter's *Leading Change*… "In a rapidly moving world, individuals and weak committees rarely have all the information needed to make good non-routine decisions. Nor do they have the credibility or the time required to convince others to make the personal sacrifices called for in implementing changes. Only teams with the right composition and sufficient trust among members can be highly effective under these circumstances."
- Guiding coalition
  - Leaders who set vision, remove impediments, make blocking the change difficult
  - Practitioners, managers, and change agents who can implement specific process changes
  - People with sufficient org credibility to be taken seriously
  - Expertise and confidence to make fast, smart decisions

Step 1: Reach the Tipping Point

- Issues
  - People naturally resist change
  - If we're changing, something must be broken

    - Our old way we liked is being challenged
- Tipping point
    - **Burning platform** (easiest) – company can’t compete, way of doing business is inadequate, survival depends on change
    - **Proactive leadership** – take a stand for a better state, people don’t see the urgency, management must continuously communicate reasons why the status quo isn’t acceptable
- Establish the vision
    - **Purpose** – objective & direction & mission (“why”)
    - **Motivation** – compelling reason to change; no job security in status quo
    - **Alignment** – empower everyone to take actions to achieve change; no constant supervision
- Communicate the benefits
    - Principle #1 - take an economic view
    - Examples: quality, time to market, engagement, productivity
    - Describe objectives to provide the fuel necessary to escape the inertia of the status quo

Step 2: Train Lean-Agile Change Agents

- Can be a mix of internal and external people at first
- Business, tech leaders, program managers, product owners, process leaders
- SAFe has courses (SAFe Program Consultant Certification)

Step 3: Train Executives, Managers, and Leaders

- SAFe has courses to teach people Lean-Agile and how to manage knowledge workers
- This is about leading rather than following the implementation

Step 4: Create a Lean-Agile Center of Excellence

- The LACE is basically a transformation group to help implement SAFe
- Seems to have overlaps with the Software Engineering Process Group from CMMI (https://en.wikipedia.org/wiki /Software_Engineering_Process_Group)
    - Cross functional (departments)
    - Led by a C-suite person
    - Permanent center of excellence for Lean-Agile learnings, communication, relentless improvement

Ch 19: Designing the Implementation

(This continues from the previous steps from Ch 18: The Guiding Coalition.)

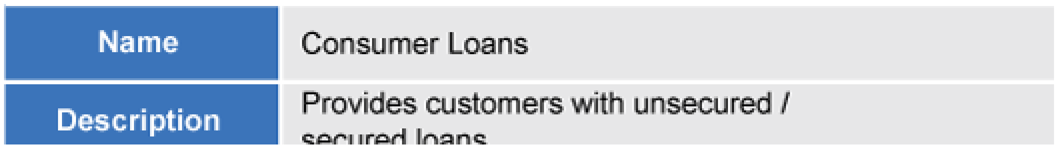
Step 5: Identify value streams and ARTs

- Agile Release Trains (ARTs) are the people and resources that build solutions that deliver value
- Triggers for flow: internal, external
- Orgs typically have two types
    - **Operational** – delivers value to the customer (define these first)
    - **Development** – builds the systems and capabilities that enable operational value streams

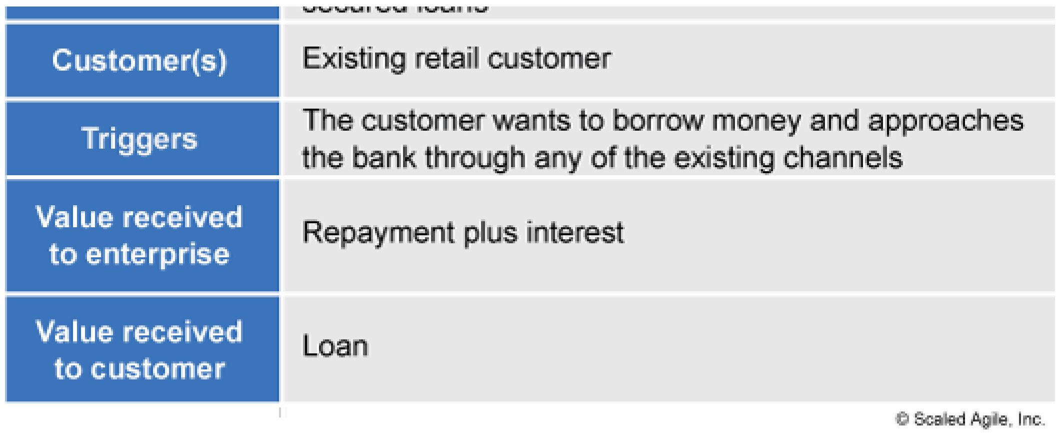
Identifying Operational Value Streams

- Products, services, solutions to develop or sell
- What’s the broader purpose of the org?
- Set of questions for stakeholders: https://www.scaledagileframework.com/identify-value-streams-and-arts

Value Stream Definition



Name	Consumer Loans
Description	Provides customers with unsecured / secured loans



Customer(s)	Existing retail customer
Triggers	The customer wants to borrow money and approaches the bank through any of the existing channels
Value received to enterprise	Repayment plus interest
Value received to customer	Loan

© Scaled Agile, Inc.

- What systems support the stream?
- Who needs to build/maintain the systems?
- Define the development streams

Realize Value Streams with ARTs

- 50-125 people
- Focused on a holistic system
- Require long-lived, stable teams to consistently deliver value
- Minimal dependencies on other ARTs
- If you need more than one ART, you have a **Solution Train**
  - Feature ARTs – optimize for flow/speed; need a system architect to maintain integrity
  - Subsystem ARTs – optimize for architectural robustness and reuse
- Some ARTs may be driven by geography or org structure; try not to do this, though
- Treat ART design as a hypothesis: balance flow with integrity, keep what works, pivot when needed

Step 6: Create the implementation plan

- Small portfolios will have obvious next streams to go after; enterprise-level portfolios will require analysis or leadership to choose

Select the First ART

Criteria:

- Leadership support
- Clear products/solutions
- Collaborating teams
- Significant challenge or opportunity

Create a Preliminary Plan for Additional ARTs and Value Streams

- Lead by the LACE (Lean Agile Center of Excellence) – I think this is Product at Lirio – which lays out the PI roadmaps of how the process will come online
- Expect change to be resisted at first; nothing is perfect

Ch 20: Implementing Agile Release Trains

(This continues from the previous steps from Ch 19: Designing the Implementation.)

Preparing for the ART Launch

Defining the ART

- The ART is a system – people, processes, management

- Business owners…
    - Responsible for business outcomes
    - Speak to the technical competence and security of the solution now and in the future
    - Participate in planning, help eliminate impediments, speak on behalf of devs/business/customer
    - Approve and defend a set of PI plans, knowing nothing is perfect
    - Help coordinate efforts of the ART with other parts of the org

Setting the Program Cadence and Launch Date

- 8-12 weeks (prefer shorter) for predictable rhythm and velocity; easier to fit into a calendar
- Cadence allows people to plan around known events, book venues, etc.
- Make the date real (deadline, planning horizon, sense of urgency, stops over-analysis)
- Pick a date based on some milestone (e.g., market window)
- If you can't pick, what's the cost of delay (is stuff so broken you can't start?)
- Aim to quickly define the backlog, socialize it, get it to a "ready state"

Training the ART Leaders and Stakeholders

- If there are people who have never done SAFe, you'll need to train them on how it works.
- (Ideally SAFe would like you to pay for their consultants to come help.) :moneybag:

Organizing the Agile Teams

- **Feature team** – user functionality, fast value delivery, end-to-end value given
- **Component team** – architectural integrity, system robustness, common components, high specialization, NFRs
- Most ARTs have a mix, but avoid having one team per layer (e.g., database, UI)

Forming the Agile Teams

- Ideally, let people self-organized with a set of minimal constraints between teams
- First time, management makes team selections based on objectives, knowledge of individuals
- Make a team roster (I've done *team charters* in the past) – team name, roles, members, locations

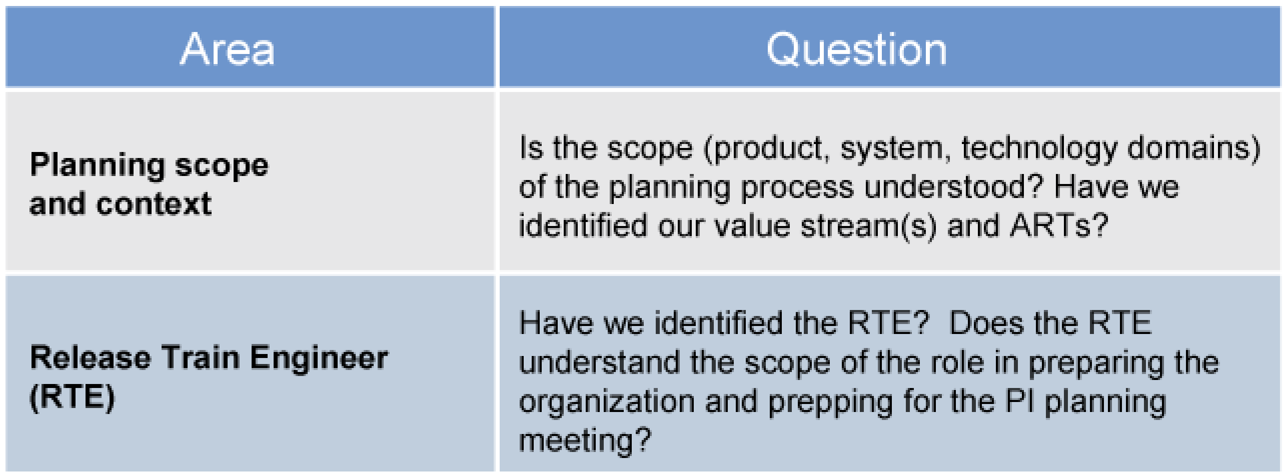
Training Product Owner and Product Managers

- POs and PMs steer the train together
- (Again, SAFe offers training :moneybag: if you want, but other Agile certs do as well.)

Training the Scrum Masters

- Roles: team leadership, improving performance, helping in PI planning, participating in Scrum of Scrums

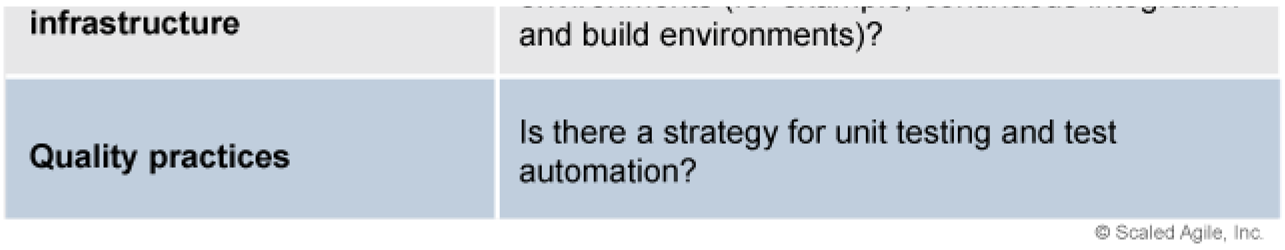
Assessing and Evolving Launch Readiness



Area	Question
Planning scope and context	Is the scope (product, system, technology domains) of the planning process understood? Have we identified our value stream(s) and ARTs?
Release Train Engineer (RTE)	Have we identified the RTE? Does the RTE understand the scope of the role in preparing the organization and prepping for the PI planning meeting?

Planning time frame, iteration and PI cadence	Have we identified the PI planning dates, the iteration cadence, and the PI cadence?
Agile teams	Does each feature/component team have an identified Scrum Master and Product Owner?
Team makeup / commitment	Does every team have dedicated team members?
Agile team attendance	Are all team members present in person or are arrangements made to involve them remotely?
Executive, Business Owner participation	Do we know who will set the business context (Business Owners) and who will present the product/solution vision (typically Product Management)?
Business alignment	Is there reasonable agreement on priorities among the Business Owner and Product Management?
Vision and program backlog	Is there a clear vision of what we are building, at least over the next few PIs? Have we identified the top 10 or so features that are the subject of the first PI?

Area	Question
System Team	Has the System Team been identified and formed?
Shared Services	Have the Shared Services (DevOps, UX, Architecture, etc.) been identified?
Other attendees	Do we know what the other key stakeholders (IT, infrastructure, etc.) should attend?
Agile project management tooling	Do we know how and where iterations, PIs, features, stories, status, etc., will be maintained?
Development	Do we understand the impact on and/or plans for environments (for example, continuous integration



infrastructure	and build environments)?
Quality practices	Is there a strategy for unit testing and test automation?

© Scaled Agile, Inc.

Preparing the Program Backlog

- This is mostly needed the first time; you need some place to capture your work (e.g., ADO, Jira)

Train Teams and Launch the ART

Training the Teams

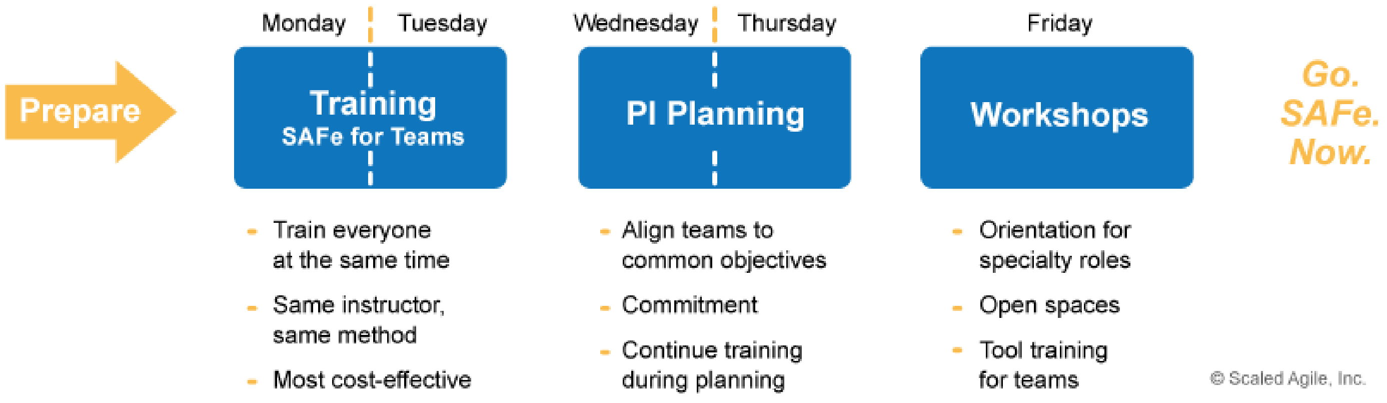
- Sure, some may understand Agile/Scrum, but they need to understand the cycles of Agile at scale
- Other topics
    - Agile dev
    - Agile manifesto
    - Core Scrum/Scrumban/Kanban/etc.
    - Backlog management

Benefits of Big-Room Planning

- Everyone is there to discuss, make decisions (no excuses of “that team is busy”)
- Collective learning activities
- Feels like a group effort, not just piecemeal or representatives only

Launch the ART

If this is the first time, SAFe prescribes something (although you may need something different)…



The First PI Planning Session

- Build confidence and enthusiasm in the new way of working
- Start to build the ART as a team-of-Agile-teams and the social network that it relies on
- Teach the teams how they can assume responsibility for planning and delivery
- Create full visibility into the mission and current context of the program
- Demonstrate the commitment of Lean-Agile Leaders to the SAFe transformation

Coaching ART execution

Implementing SAFe does not make you agile; you need active coaching in the ART to encourage learning and growth. Empower people in their new way working

Coaching within the ART

- Build and maintain the vision and roadmap and program backlog
- PMs, system architects, and RTEs
- System-level integration
- System demo
- Scrum of Scrums (or PO/ART sync meetings)
- Facilitate Inspect & Adapt events
- DevOps culture, CD pipelines
- Release management
- Training

Coaching the Teams

- Help plan, execute, review, and retro the first iterations
- Coach new Scrum Masters and POs
- Initiate and support Agile software engineering practices
- Help establish infrastructure, DevOps, culture of continuous delivery
- Encourage and support Communities of Practice (CoPs)

Inspect and Adapt

- This lets you know whether what you planned aligned with what you expected
- How well is SAFe adoption working?

Ch 22: Sustaining and Improving

Foster Relentless Improvement and the Lean-Agile Mindset

- Leadership and relentless improvement are inseparable
- Ongoing leadership training
- Continuing role of the Lean-Agile Center of Excellence (LACE)
    - Initially they help get SAFe implemented
    - Afterward, they continue to feed and improve the system
- Communities of Practices
    - Organize people with different skills around a value stream, subject domain, or area of common interest
    - (Spotify calls these *guilds*)

Implement Agile HR Process

1. Embrace a modern talent contract that acknowledges the need for value, autonomy, and empowerment
2. Foster continuous engagement with business and technical missions
3. Hire people for Agile attitude, team orientation, and cultural fit
4. Eliminate annual performance reviews; use continuous, iterative performance feedback and evaluation
5. Eliminate demotivating individual financial incentives; pay people enough to take money off the table
6. Support meaningful, impactful, and continuous learning and growth

Advance Program Execution and Servant Leadership Skills

- Targets: Scrum Masters, RTEs
- SAFe offers formal training (for a fee)

Measure and Take Action

- Ch 10: Inspect and Adapt
- Ch 17: Lean Governance
- https://www.scaledagileframework.com/metrics/
    - This also has some self-assessments

Improve Agile Software Engineering Competencies

- Use the Innovation and Planning iteration to focus on these topics (e.g., CI, test automation).
- Some topics that seem a bit academic or “dressed up for business”
    - Model-Based Systems Engineering (MBSE)
    - Set-Based Design (SBD)

Focus on Agile Architecture

- Big Up-Front Design (BFUD) doesn’t work anymore; you have to change the engine while you’re driving
- Agile Architecture Center of Practice topics
    - Review SAFe principles of Agile architecture
    - Identify enabler epics and capabilities needed to evolve the solution architecture
    - Identify methods of splitting architectural epics into enabler capabilities and features for incremental implementation
    - Establish the decision-making framework and policies for architectural governance and capacity allocation
    - Identify relevant NFRs

Improve DevOps and CD Capability

- Once ARTs are launched, value streams operate better and bottlenecks/impediments become more obvious
- See Ch 9: Executing the Program Increment and other DevOps practices; CoPs can help with ownership as well

Reduce Time-to-Market with Value Stream Mapping

- Each value stream provides an identifiable and measurable flow of value to the customer

Here’s how to do it: (I think this is the feedback loop from Eliyahu Goldratt’s *The Goal* from 1984!)

1. From request to release…. map all steps, value-added times, hand-offs, delays
2. Identify the most frequent sources of delays and hand-offs in the system
3. Find the biggest delay, then use root-cause analysis. Backlog a solution to this.
4. Implement the backlog items.
5. Measure again, then repeat the process.

Ch 21: Launching More ARTs and Value Streams

Introduction

(This continues from the previous step of Ch 20: Implementing Agile Release Trains.)

Launch More ARTs and Value Streams

Launch More ARTs

- LACE and other stakeholders repeat the same steps they used to launch existing ARTs (prepare, train teams, coach)
- Use the same care as you did the first time; don’t fall into the “everyone knows how to do this now” mindset

Implement Large Solution Roles, Artifacts, and Events

- **Solution train** coordinates ARTs
- See the chapters on Large Solution SAFe for specifics

Launch More Value Streams

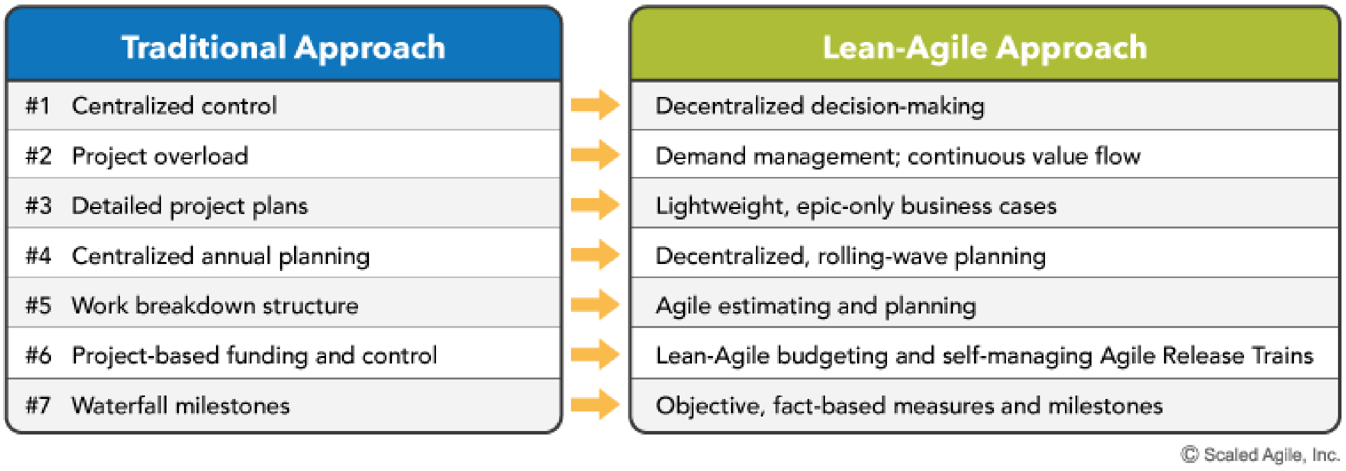
- What's likely changed… businesses, operating units, countries, chains of command
- Look at each value stream and see where to build the ART (see Ch 20: Implementing Agile Release Trains)
- Follow principle #6 (see Ch 4: SAFe Principles) about visualizing WIP

The SAFe Implementation Railway

- See case study from Northwestern Mutual: https://www.scaledagileframework.com/launch-more-arts-and-value-streams/

Extend to the Portfolio

- Legacy challenges
    - Demand > capacity
    - Project-based funding, cost-accounting friction, and overhead
    - No understanding of how to capitalize expenses in an Agile business
    - Overly detailed business cases based on speculative, lagging ROI projections
    - Strangulation by the iron triangle (scope, cost, time)
    - Traditional supplier management coordination (lowest cost > highest life-cycle)
    - Phase-gated approval processes that discourage incremental delivery



Traditional Approach		Lean-Agile Approach
#1 Centralized control	→	Decentralized decision-making
#2 Project overload	→	Demand management; continuous value flow
#3 Detailed project plans	→	Lightweight, epic-only business cases
#4 Centralized annual planning	→	Decentralized, rolling-wave planning
#5 Work breakdown structure	→	Agile estimating and planning
#6 Project-based funding and control	→	Lean-Agile budgeting and self-managing Agile Release Trains
#7 Waterfall milestones	→	Objective, fact-based measures and milestones

© Scaled Agile, Inc.

- Align value streams to the enterprise strategy
- Establish enterprise value flow – need epic owners and enterprise architecture
- Implement lean financial management (see Ch 17: Lean Governance)
- Agile portfolio demand to capacity and agile forecasting – a system operating in a state of overload will deliver far less
- Evolve leaner and more objective governance practices
- Foster a learner approach to supplier and customer relationships