SAFe®
REFERENCE GUIDE
Scaled Agile Framework® for Lean Software and Systems Engineering

SAFe® 4.0 for Lean Software and Systems Engineering

Dean Leffingwell
with Alex Yakyma, Richard Knaster,
Drew Jemilo, and Inbar Oren
Contents

Preface ............................................................................................................................................................ vii

Acknowledgments ....................................................................................................................................... ix
  Scaled Agile Framework Contributors.................................................................................................... ix
  SAFe Community Contributors ............................................................................................................. x
  Additional Acknowledgments ................................................................................................................ x

Introduction to the Scaled Agile Framework (SAFe) ................................................................................. 1

Part 1: The SAFe Foundation ..................................................................................................................... 9
  Lean-Agile Leaders ................................................................................................................................ 11
  Communities of Practice ......................................................................................................................... 17
  SAFe Core Values ................................................................................................................................... 21
  Lean-Agile Mindset ................................................................................................................................ 27
  SAFe Principles ....................................................................................................................................... 35
  Implementing 1-2-3 .............................................................................................................................. 37

Part 2: The SAFe Principles ..................................................................................................................... 43
  #1 – Take an economic view.................................................................................................................. 45
  #2 – Apply systems thinking ................................................................................................................. 51
  #3 – Assume variability; preserve options ............................................................................................. 55
  #4 – Build incrementally with fast, integrated learning cycles ............................................................... 57
  #5 – Base milestones on objective evaluation of working systems ..................................................... 59
  #6 – Visualize and limit WIP, reduce batch sizes, and manage queue lengths .................................. 61
  #7 – Apply cadence, synchronize with cross-domain planning ............................................................ 63
  #8 – Unlock the intrinsic motivation of knowledge workers ................................................................. 67
  #9 – Decentralize decision-making .................................................................................................... 71
# Part 3: The Team Level

- Introduction to the Team Level ................................................................. 75
- Agile Teams ............................................................................................ 77
- Product Owner ...................................................................................... 83
- Scrum Master ......................................................................................... 89
- ScrumXP ................................................................................................. 93
- Team Kanban .......................................................................................... 99
- Team Backlog ........................................................................................ 105
- Iterations ............................................................................................... 109
- Iteration Planning .................................................................................. 113
- Iteration Execution ................................................................................ 119
- Team Demo ............................................................................................ 125
- Iteration Retrospective ......................................................................... 129
- Stories .................................................................................................... 133
- Iteration Goals ...................................................................................... 143
- Built-In Quality ...................................................................................... 147

# Part 4: The Program Level

- Introduction to the Program Level .......................................................... 155
- Agile Release Train ................................................................................ 159
- Release Train Engineer and Value Stream Engineer ............................. 167
- System and Solution Architect/Engineering ........................................ 171
- Product and Solution Management ....................................................... 177
- WSJF (Weighted Shortest Job First) ...................................................... 183
- Program and Value Stream Kanban ...................................................... 187
- Program and Value Stream Backlogs .................................................. 193
- Nonfunctional Requirements ................................................................. 199
- Program Increment ............................................................................... 207
- PI Planning ............................................................................................ 213
- Business Owners .................................................................................. 221
- PI Objectives ......................................................................................... 225
- System Demo ....................................................................................... 233
Contents

Features and Capabilities..................................................................................................................237
Enablers..................................................................................................................................................243
Innovation and Planning Iteration......................................................................................................249
Inspect and Adapt ...............................................................................................................................253
Develop on Cadence, Release Any Time ..........................................................................................259
Architectural Runway..........................................................................................................................265

Part 5: The Spanning Palette .............................................................................................................271
DevOps ..................................................................................................................................................273
System Team .........................................................................................................................................279
Release Management .........................................................................................................................283
Shared Services ..................................................................................................................................287
User Experience (UX) .........................................................................................................................291
Vision ...................................................................................................................................................295
Roadmap ...............................................................................................................................................301
Metrics ...................................................................................................................................................307
Milestones ............................................................................................................................................323
Releases ...............................................................................................................................................331

Part 6: The Value Stream Level .........................................................................................................337
Introduction to the Value Stream Level .............................................................................................339
Value Stream Coordination .................................................................................................................343
Economic Framework ..........................................................................................................................347
Solution Intent ........................................................................................................................................351
Model-Based Systems Engineering ....................................................................................................359
Set-Based Design ..................................................................................................................................365
Agile Architecture ...............................................................................................................................371
Solution Demo .........................................................................................................................................379
Pre- and Post-PI Planning ....................................................................................................................383
Supplier ..................................................................................................................................................389
Customer ...............................................................................................................................................395
Solution ..................................................................................................................................................401
Solution Context ......................................................................................................................................405
Part 7: The Portfolio Level ................................................................. 411
  Introduction to the Portfolio Level ............................................. 413
  Enterprise ................................................................................. 417
  Strategic Themes .................................................................... 423
  Program Portfolio Management ................................................... 429
  Epic Owners ............................................................................ 435
  Enterprise Architect ................................................................ 439
  Portfolio Kanban ..................................................................... 443
  Portfolio Backlog .................................................................... 449
  Budgets ................................................................................... 453
  CapEx and OpEx ...................................................................... 463
  Value Streams .......................................................................... 473
  Epics ....................................................................................... 483

Part 8: Guidance ................................................................. 489
  Continuous Integration .............................................................. 491
  Test-First ................................................................................ 497
  Agile Contracts ........................................................................ 503
  Glossary of SAFe Terms ............................................................ 511

Bibliography ............................................................................... 523

Index ........................................................................................ 529
On behalf of the entire Scaled Agile, Inc., team and the SAFe contributors, it is my personal pleasure to introduce the SAFe 4.0 Reference Guide.

SAFe is an online, freely revealed knowledge base of proven success patterns for implementing Lean-Agile software and systems development at enterprise scale. It provides comprehensive guidance for work at the enterprise Portfolio, Value Stream, Program, and Team levels.

Why SAFe?

The world’s economy, and the health and welfare of society as a whole, is increasingly dependent on software and systems. In support of this need, systems builders are creating increasingly complex software and cyber-physical systems of unprecedented scope and complexity with requirements for utility and robustness exceeding those that have come before them.

The methods that systems builders use to create these systems must keep pace with this larger mandate. However, the assumptive, one-pass, stage-gated, waterfall methods of the past are not scaling to the new challenge. New development methods are needed. Agile shows the greatest promise, but it was developed for small teams and, by itself, does not scale to the needs of larger enterprises and the systems they create. What’s needed is a new way of working, one that applies the power of Agile but leverages the more extensive knowledge pools of systems thinking and Lean product development. The Scaled Agile Framework (SAFe) is one such approach.

SAFe is provided by Scaled Agile, Inc., where our core belief is simple: Better systems and software make the world a better place. Our mission is to assist those who build these systems through development and publication of the SAFe framework, as well as accompanying certification, training, and courseware. As case studies on the Scaled Agile Framework website (www.scaledagileframework.com) show, many enterprises—large and small—are getting outstanding business benefits from applying SAFe. These typically include:

- 20 – 50% increase in productivity
- 50%+ increases in quality
- 30 – 75% faster time to market
- Measurable increases in employee engagement and job satisfaction
As you can imagine, with results like those, SAFe is spreading rapidly around the world. The majority of Fortune 100 U.S. companies have certified SAFe practitioners and consultants already on site, as do an increasing percentage of the Global 1000 enterprises. SAI also has an extensive network of more than 80 global partners providing consulting and implementation services in almost every region of the world. And while every business situation is unique, we have found that the straightforward Implementation 1-2-3 strategy always delivers results.

Our commitment is to continuously evolve SAFe to provide value to the industry—better systems, better business outcomes, better daily lives for the people who build the world's most important new systems—but only you, the adopters and practitioners, can tell us whether or not we have accomplished that. As we are fond of saying, “without you, SAFe is just a website.”

**Why a Reference Guide?**

The SAFe website is quite comprehensive, containing hundreds of pages of guidance for the various roles, responsibilities, activities, and artifacts that constitute SAFe, along with the foundational elements of values, Lean-Agile mindset, principles, and practices. This Reference Guide is intended to provide a hard-copy or e-copy companion to help you understand and apply SAFe, with the goal of helping you accomplish your mission of building better systems.

**More on SAFe**

For more on SAFe, please browse the site, read the blog, watch the “updates” field, and follow us on Twitter (@ScaledAgile), where we will notify you of new developments. Also, click on the site's “Presentations & Downloads” tab to find free posters for the Big Picture, the House of Lean, and SAFe Lean-Agile Principles. You'll also find SAFe videos and recorded webinars, free presentations, and more. Finally, be sure to check out our corporate site www.ScaledAgile.com. Even better, attend a Training and Certification course (www.scaledagile.com/which-course); perhaps I will see you there. Stay SAFe!

—*Dean Leffingwell and the Scaled Agile Team*
Acknowledgments

Scaled Agile Framework Contributors

Alex Yakyma, SAFe Fellow and Principal Consultant
Alex is a SAFe methodologist, trainer, and principal consultant who has been involved with the development and field implementations of the Scaled Agile Framework since its inception. Alex's broad prior experience as an engineer, development manager, and program manager in highly distributed multicultural environments provides the experience needed to assist enterprises with improving their system development capabilities at the Program, multi-program, and Portfolio levels. Alex has published a number of articles and white papers on Agile and Lean and is the author of Pacific Express, a novella about launching an Agile Release Train.

Drew Jemilo, SAFe Fellow and Principal Consultant
Drew is a principal contributor to the Scaled Agile Framework, a consultant, and an instructor. Drew met Dean Leffingwell in early 2009 when he was developing a scaled Agile methodology for a management consulting company to bridge their strategic business framework with Agile. Since then, they have worked together with global clients to synchronize distributed teams using Agile Release Trains in the United States, Europe, and India.

Richard Knaster, SAFe Fellow and Principal Consultant
Richard has more than 25 years’ experience in software development in roles ranging from developer to executive and has been involved in Agile for more than a decade. Prior to joining Scaled Agile, Inc., Richard worked at IBM, where his career spanned from product line management (PPM domain) and professional services to chief methodologist, Agile and Lean. Richard is a certified IBM Thought Leader and an Open Group Distinguished IT Specialist. He is also a certified SPC, PSM, Agile Certified Practitioner, PMI, and a contributor to the Disciplined Agile Delivery framework and PMI Portfolio/Program Management standards.

Inbar Oren, SAFe Fellow and Principal Consultant
Inbar has more than 20 years’ experience in the high-tech market. For more than a decade, he has been helping development organizations—in both software and integrated systems—improve results by adopting Lean-Agile best practices. Previous clients include Cisco, Woolworth, Amdocs, Intel, and NCR. Inbar's current focus is on working with leaders at the Program, Value Stream, and Portfolio levels to help them bring the most out of their organizations and build new processes and culture.
SAFe Community Contributors

We are also indebted to those SAFe Program Consultant Trainers (SPCTs) and SAFe Program Consultants (SPCs) who are doing the hard work of applying the framework in various enterprises every day. Many have contributed indirectly in discussions, certification workshops, LinkedIn forums, and more. More specifically, the following individuals have directly provided content that is included either here or in Guidance articles on the Scaled Agile Framework website (www.scaledagileframework.com).

- **Harry Koehnemann, SPCT** – Special contributor to SAFe for Lean Systems Engineering and 4.0 systems engineering content
- **Ken France, SPCT** – Guidance article: “Mixing Agile and Waterfall Development in the Scaled Agile Framework”
- **Scott Prugh, SPC** – Guidance article: “Continuous Delivery”
- **Eric Willeke, SPCT** – Guidance articles: “Role of PI Objectives,” “A Lean Perspective on SAFe Portfolio WIP Limit”
- **Jennifer Fawcett, SAFe Fellow and Principal Consultant** – Product Manager and Product Owner contribution and focus
- **Colin O’Neill, SPCT** – SAFe 1.0 – 2.5 contributor
- **Gareth Evans, SPCT** – Guidance article: “Lean Software Development in SAFe”
- **Gillian Clark, SPCT** – Guidance article: “Lean Software Development in SAFe”
- **Maarit Laanti, SPC** – “Lean-Agile Budgeting” guidance and white paper
- **Steven Mather, SPC** – SAFe 2.0 glossary draft
- **Al Shalloway, SPCT** – Concept development and community support

Additional Acknowledgments

The Contributors to Agile Software Requirements

The initial concepts behind the framework were first documented in the 2007 text *Scaling Software Agility: Best Practices for Large Enterprises*, by Dean Leffingwell. But the framework itself was first documented in Dean's 2011 book *Agile Software Requirements: Lean Requirements for Teams, Programs, and the Enterprise* (ASR), so it's appropriate to repeat and update the book acknowledgments here.

Thanks to the ASR reviewers, Gabor Gunyho, Robert Bogetti, Sarah Edrie, and Brad Jackson. Don Reinertsen provided permission to use elements of his book, *The Principles of Product Development Flow*. Thanks to my Finnish collaborators: Juha-Markus Aalto, Maarit Laanti, Santeri Kangas, Gabor Gunyho, and Kuan Eeik Tan. Alistair Cockburn, Don Widrig, Mauricio Zamora, Pete Behrens, Jennifer Fawcett, and Alexander Yakyma contributed directly to book content. Even that list is not exhaustive; many others—Mike Cottmeyer, Ryan Shriver, Drew Jemilo, Chad Holdorf, Keith Black, John Bartholomew, Chris Chapman, Mike Cohn, Ryan Martens, Matthew Balchin, and Richard Lawrence—contributed words, thoughts, or encouragement.
A Special Acknowledgment to the Agile Thought Leaders
Of course, SAFe stands on the shoulders of many who came before us, particularly the Agile thought leaders who created the industry movement. It starts with the signers of the Agile Manifesto and continues with those outspoken thought leaders who have helped move the industry toward the new paradigm. The following have contributed most directly to our understanding of Agile development: Kent Beck, Alistair Cockburn, Ron Jeffries, Mike Cohn, David Anderson, Jeff Sutherland, Martin Fowler, Craig Larman, Ken Schwaber, Scott Ambler, and Mary and Tom Poppendieck. Still others are acknowledged in the Bibliography.

A Special Acknowledgment to the Lean Leaders
In extending Agile to the enterprise and developing the broader Lean-Agile paradigm, we are fortunate to stand on the shoulders of Lean thought leaders as well, including Don Reinertsen, Jeffrey Liker, Taichi Ohno, Eli Goldratt, Dr. Alan Ward, Jim Sutton, Michael Kennedy, Dantar Oosterwal, Steve Womack, and Daniel Jones. Still others are acknowledged in the Bibliography.

And to W. Edwards Deming
Finally, where would we be without the seminal works of W. Edwards Deming, to whom we perhaps owe the deepest gratitude of all? He was a visionary and systems thinker, whose tireless quest for the underlying truths and unwavering belief in people and continuous improvement led to a set of transformational theories and teachings that changed the way we think about quality, management, and leadership.
Introduction to the Scaled Agile Framework (SAFe)

The Scaled Agile Framework (SAFe) is a freely revealed knowledge base of proven, integrated patterns for enterprise-scale Lean-Agile development. It is scalable and modular, allowing each organization to apply it in a way that provides better business outcomes and happier, more engaged employees.

SAFe synchronizes alignment, collaboration, and delivery for large numbers of Agile Teams. It supports both software solutions and complex cyber-physical systems that require thousands of people to create and maintain. SAFe was developed in the field, based on helping Customers solve their most challenging scaling problems. SAFe leverages three primary bodies of knowledge: Agile development, Lean product development and flow, and systems thinking.

Overview

The SAFe website (www.scaledagileframework.com) provides comprehensive guidance for scaling development work across all levels of an enterprise. SAFe's interactive “Big Picture” (Figure 1) provides a visual overview of the framework. Each icon on the website is selectable, navigating the user to an article that provides extensive guidance on the topic area, along with links to related articles and further information.

The Big Picture has two views. The default “3-level view” (below left) is well suited for solutions that require a modest number of Agile Teams. The “4-level view” (below right) supports those building large solutions that typically require hundreds or more practitioners to construct and maintain.

Figure 1. Big Picture: 3-level and 4-level SAFe
SAFe provides three, and optionally four, organization levels, as well as a foundation, as follows:

- **Team Level** – SAFe is based on Agile Teams, each of which is responsible for defining, building, and testing stories from their backlog. Teams employ Scrum or Kanban methods, augmented by quality practices, to deliver value in a series of synchronized, fixed-length iterations.

- **Program Level** – SAFe teams are organized into a virtual program structure called the “Agile Release Train” (ART). Each ART is a long-lived, self-organizing team of 5 to 12 Agile Teams—along with other stakeholders—that plan, commit, execute, inspect and adapt, and deliver solutions together.

- **Value Stream Level** – The optional value stream level supports the development of large and complex solutions. These solutions require multiple, synchronized ARTs, as well as stronger focus on solution intent and solution context. Suppliers and additional stakeholders contribute as well.

- **Portfolio Level** – The portfolio level organizes and funds a set of value streams. The portfolio provides solution development funding via Lean-Agile budgeting and provides necessary governance and value stream coordination.

- **Foundation Layer** – The foundation layer holds various additional elements that support development. Elements include guidance for Lean-Agile Leaders, communities of practice, core values, the Lean-Agile mindset, the nine Lean-Agile principles that guide SAFe, and an overview of implementation strategy.

### Foundation Layer

The foundation layer of SAFe (the shadow backdrop on the big picture) contains the aspects of SAFe that are critical, necessary, and supportive of value delivery, but are not specific practices. This layer contains the following:

- **Lean Agile Leaders** – The ultimate responsibility for the success of the enterprise, and thereby any significant change to the way of working, lies with management. To this end, SAFe describes a new style of leadership, one that is exhibited by SAFe's Lean-Agile Leaders.

- **Communities of Practice** – The Lean approach to aligning around Value Streams typically causes the Lean enterprise to pivot from a functional organization to a more flexible and adaptive line-of-business approach. In response, SAFe also supports communities of practice, informal groups of team members and other experts who share practical, functional knowledge in one or more relevant domains.

- **Core Values** – There are four primary core values that help make SAFe effective: Alignment, Built-in Quality, Transparency, and Program Execution.
• **Lean-Agile Mindset** – SAFe Lean-Agile Leaders are lifelong learners and teachers who understand and embrace Lean and Agile principles and practices, and teach them to others. To achieve that effectively, leaders must first be trained in, and then become trainers of, these leaner ways of thinking and operating. This mindset is exhibited in part by the House of Lean and the Agile Manifesto.

• **Lean-Agile Principles** – SAFe’s practices are grounded on nine fundamental principles that have evolved from Agile principles and methods, Lean product development, systems thinking, and observation of successful enterprises. These are:
  - #1 – Take an economic view
  - #2 – Apply systems thinking
  - #3 – Assume variability; preserve options
  - #4 – Build incrementally with fast, integrated learning cycles
  - #5 – Base milestones on objective evaluation of working systems
  - #6 – Visualize and limit WIP, reduce batch sizes, and manage queue lengths
  - #7 – Apply cadence, synchronize with cross-domain planning
  - #8 – Unlock the intrinsic motivation of knowledge workers
  - #9 – Decentralize decision-making

• **Implementing 1-2-3** – Based on the learnings from hundreds of SAFe implementations, a basic “Implementing SAFe 1-2-3” pattern for successfully adopting SAFe has emerged. The pattern is, first, train implementers and Lean-Agile change agents (SPCs). In turn, these external or internal consultants can then train all executives, managers, and leaders. After this, SPCs can train teams and launch Agile Release Trains.

**Team Level**

The team level provides an organization, artifact, role, and process model for the activities of Agile Teams, as illustrated in Figure 2.

![Figure 2. SAFe team level](image-url)
All SAFe teams are part of one Agile Release Train (ART)—the central construct of the program level. Each Agile Team is responsible for defining, building, and testing stories from their team backlog in a series of fixed-length iterations, using common iteration cadence and synchronization to align with other teams, so that the entire system is iterating.

Teams use Scrum or Team Kanban, along with the Built-in Quality practices, to deliver working software every two weeks. The system demo creates a routine “pull event,” which pulls the effort of the different teams together, bringing forward the hard work of integration and testing that phase-gated models often leave until too late in the life cycle.

Each team has five to nine members and includes all the roles necessary to build a quality increment of value in each iteration. Roles include the Scrum Master, Product Owner, dedicated individual contributors, and any specialty resources the team needs to deliver value.

A summary of this level is provided in the “Introduction to the Team Level” overview article.

**Program Level**

The heart of SAFe is the program level, illustrated in Figure 3, which revolves around the organization called the “Agile Release Train,” and which incorporates the team level by reference.

SAFe program level teams, roles, and activities are organized around the ART metaphor, a team of Agile Teams that delivers a continuous flow of incremental releases of value.
ARTs are virtual organizations formed to span functional boundaries, eliminate unnecessary handoffs and steps, and accelerate the delivery of value via implementation of SAFe Lean-Agile principles and practices.

While it is called the “program level,” ARTs are generally very long-lived and therefore have a more persistent self-organization, structure, and mission than a traditional “program,” which more classically has a start and an end date, as well as temporarily assigned resources. It is the long-lived, knowledge acquiring, flow-based, and self-organizing nature of the ART that powers the SAFe portfolio.

Value in SAFe is delivered by Agile Release Trains, each of which realizes a portion of a value stream (or, in some cases, the entire value stream). They deliver value incrementally in program increments (PIs) of 8 to 12 weeks in duration; each PI is a multiple-iteration timebox during which a significant, valuable increment of the system is developed. Each ART is composed of 5 to 12 Agile Teams (50 – 125+ people) and includes the roles and infrastructure necessary to deliver fully tested, working, system-level solutions. Many release trains are virtual, spanning organizational and geographic boundaries; others follow a line of business or product line management reporting structure.

A summary of this level is provided in the “Introduction to the Program Level” overview article.

**The Spanning Palette**

There are a number of additional icons indicated on this level; they are located at the conjunction of the value stream and program level on the Big Picture. This is called the “spanning palette” and is illustrated in Figure 4.

![Figure 4. The spanning palette](image-url)

Each of these artifacts and roles contributes to the ART and program level, as described in the “Vision,” “Roadmap,” “Metrics,” “Milestones,” “Releases,” “DevOps,” “System Team,” “Release Management,” “Shared Services,” and “User Experience” articles. However, these elements also “span” the levels because many of them are also useful at the other levels.
Value Stream Level

The value stream level is optional in SAFe. Enterprises that build systems that are largely independent, or that can be built with a few hundred practitioners, may not need these constructs, and in that case the portfolio can operate with the 3-level view. Even then, however, those are far from trivial systems, and the constructs at the value stream level can be used in 3-level SAFe as needed.

The value stream level helps enterprises that face the largest systems challenges: those building large-scale, multidisciplinary software and cyber-physical systems. Building such solutions in a Lean-Agile manner requires additional constructs, artifacts, and coordination. The constructs of the value stream level are illustrated in Figure 5.

This level contains an economic framework, intended to provide financial boundaries for value stream and ART decision-making; solution intent as a repository for intended and actual solution behavior; solution context, which describes the way the solution fits in the deployment environment; and capabilities, describing the larger behaviors of the solution.

Like the program level, the value stream level is organized around program increments, which are synchronized across all the ARTs in the value stream. It provides for cadence and synchronization of multiple ARTs and Suppliers, including pre- and post-PI planning meetings and the solution demo. It also provides additional roles, specifically Solution Management, Solution Architect/Engineering, and the Value Stream Engineer.

A summary of this level may be found in the “Introduction to the Value Stream Level” overview article.

Portfolio Level

The SAFe portfolio is the highest level of concern in SAFe. As illustrated in Figure 6, each SAFe portfolio has the value streams, people, and processes necessary to provide funding and governance for the products, services, and solutions required to fulfill the overall business strategy.
It provides the basic constructs for organizing the Lean-Agile Enterprise around the flow of value via one or more value streams, each of which develops the systems and solutions necessary to meet the strategic intent. The portfolio level encapsulates these elements and also provides the basic budgeting and other governance mechanisms that are necessary to ensure that the investment in the value streams provides the returns necessary for the enterprise to meet its strategic objectives.

The portfolio has a bidirectional connection to the business. One direction provides the strategic themes that guide the portfolio to the larger, and changing, business objectives. The other direction indicates a constant flow of portfolio context back to the enterprise.

The primary elements of the portfolio are value streams (one or more), each of which provides funding for the people and other resources necessary to build the solutions that deliver the value. Each value stream is a long-lived series of system definition, development, and deployment steps used to build and deploy systems that provide a continuous flow of value to the business or Customer. Program Portfolio Management represents the stakeholders who are accountable to deliver the business results.

A summary of this level may be found in the “Introduction to the Portfolio Level” overview article.
This page intentionally left blank
Part 1
The SAFe Foundation
This page intentionally left blank
Lean-Agile Leaders

A leader is one who knows the way, goes the way, and shows the way.
—John C. Maxwell

Abstract

The philosophy of SAFe is simple: As the enabler for the teams, the ultimate responsibility for adoption, success, and ongoing improvement of Lean-Agile development lies with the Enterprise’s existing managers, leaders, and executives. Only they can change and continuously improve the systems in which everyone operates. To achieve this, leaders must be trained, and become trainers, in these leaner ways of thinking and operating. Many need to offer a new style of leadership, one that truly teaches, empowers, and engages individuals and teams to reach their highest potential.

While some of these management roles and titles do not appear specifically on the Big Picture, they serve a critical function nonetheless by providing the personnel, resources, management, direction, and support necessary to help the enterprise achieve its mission. This article describes the principles of these Lean-Agile Leaders.

Details

SAFe Lean-Agile Leaders are lifelong learners and teachers who help teams build better systems through understanding and exhibiting the Lean-Agile Mindset, SAFe Principles, and systems thinking. Such leaders exhibit the behaviors below.

#1 – Lead the Change

The work of steering an organization toward Lean and Agile behaviors, habits, and results cannot be delegated. Rather, Lean-Agile Leaders exhibit urgency for change, communicate the need for the change, build a plan for successful change, understand and manage the change process, and address problems as they come up. They have knowledge of organizational change management and take a systems view with respect to implementing the transformation.
#2 – Know the Way; Emphasize Lifelong Learning
Create an environment that promotes learning. Encourage team members to build relationships with Customers and Suppliers and expose them to other world views. Strive to learn and understand new developments in Lean, Agile, and contemporary management practices. Create and foster formal and informal groups for learning and improvement. Read voraciously from the recommended reading list and on other topics. Share selected readings with others and sponsor book club events for the most relevant texts.

Allow people to solve their own problems. Help them identify a given problem, understand the root causes, and build solutions that will be embraced by the organization. Support individuals and teams when they make mistakes, otherwise learning is not possible.

#3 – Develop People
Employ a Lean leadership style, one that focuses on developing skills and career paths for team members rather than on being a technical expert or coordinator of tasks. Create a team jointly responsible for success. Learn how to solve problems together in a way that develops people's capabilities and increases their engagement and commitment. Respect people and culture.

#4 – Inspire and Align with Mission; Minimize Constraints
Provide mission and vision, with minimum specific work requirements. Eliminate demotivating policies and procedures. Build Agile Teams and trains organized around value. Understand the power of self-organizing, self-managing teams. Create a safe environment for learning, growth, and mutual influence. Build an Economic Framework for each Value Stream and teach it to everyone.

#5 – Decentralize Decision-Making
(See “SAFe Principle #9” for further discussion.)
Establish a decision-making framework. Empower others by setting the mission, developing people, and teaching them to problem-solve. Take responsibility for making and communicating strategic decisions—those that are infrequent, long lasting, and have significant economies of scale. Decentralize all other decisions.

#6 – Unlock the Intrinsic Motivation of Knowledge Workers
(See “SAFe Principle #8” for further discussion.)
Understand the role that compensation plays in motivating knowledge workers. Create an environment of mutual influence. Eliminate any and all management by objectives (MBOs) that cause internal competition. Revamp personnel evaluations to support Lean-Agile principles and values. Provide purpose and autonomy; help workers achieve mastery of new and increasing skills.
Role of the Development Manager

As an instantiation of the principles of Lean and Agile development, SAFe emphasizes the values of nearly autonomous, self-organizing, cross-functional teams and Agile Release Trains. This supports a leaner management infrastructure, with more empowered individuals and teams and faster, local decision-making. Traditional, day-to-day employee instruction and activity direction is no longer required.

However, all employees still need someone to assist them with career development; set and manage expectations and compensation; and provide the active coaching they need to advance their technical, functional, individual, and team skills and career goals. They also have a right to serve as an integral member of a high-performing team.

In addition, self-organizing ARTs do not fund themselves or define their own mission. That remains a management responsibility, as it is an element of implementation of strategy.

Much of this responsibility traditionally falls to the traditional role of the development manager, and the adoption of Lean-Agile development does not abrogate their responsibilities. However, in SAFe these responsibilities fall to those who can adapt, thrive, and grow in this new environment.

Responsibilities

The development manager (or engineering manager for system development) is a manager who exhibits the principles and practices of Lean-Agile leadership as described above. Further, the manager has personal responsibility for the coaching and career development of direct reports, takes responsibility for eliminating impediments, and actively evolves the systems in which all knowledge workers operate. They have final accountability for effective value delivery as well. A summary of responsibilities is highlighted below.

Personnel and Team Development

- Attract, recruit, and retain capable individuals
- Build high-performing teams; establish mission and purpose for individuals and teams
- Perform career counseling and personal development
- Listen and support teams in problem identification, root cause analysis, and decision-making
- Participate in defining and administering compensation, benefits, and promotions
- Eliminate impediments and evolve systems and practices in support of Lean-Agile development
- Take subtle control in assignment of individuals to teams; address issues that teams cannot unblock; make personnel changes where necessary
• Evaluate performance, including team input; provide input, guidance, and corrective actions
• Serve as Agile coach and advisor to Agile Teams
• Remain close enough to the team to add value and to be a competent manager; stay far enough away to let them problem-solve on their own

Program Execution
• Help in building Agile Milestones and Roadmaps, as well as the building plans that enable them
• Help develop, implement, and communicate the economic framework
• Participate in Inspect and Adapt workshops
• Protect teams from distractions and unrelated or unnecessary work
• Assist the Release Train and Value Stream Engineers with PI Planning readiness and Pre- and Post-PI Planning activities
• Participate in PI planning, System Demo, and Solution Demo
• Build partnerships with Suppliers, subcontractors, consultants, partners, and internal and external stakeholders
• Provide other resources as necessary for teams and ARTs to successfully execute their Vision and roadmap

Alignment
• Work with Release Train and Value Stream Engineers and system stakeholders to help ensure alignment and effective execution of Strategic Themes
• Work with the System Architect/Engineer, Product Managers, and Product Owners to establish clear content authority
• Continuously assist in aligning teams to the system mission and vision
• Help ensure the engagement of Business Owners, Shared Services, and other stakeholders

Transparency
• Create an environment where the facts are always friendly
• Provide freedom and safety so individuals and teams are free to innovate, experiment, and even fail on occasion
• Communicate openly and honestly with all stakeholders
• Keep backlogs and information radiators fully visible to all
• Value productivity, quality, transparency, and openness over internal politics
**Built-in Quality**

- Understand, teach, or sponsor technical skills development in support of high-quality code, components, systems, and Solutions
- Foster Communities of Practice
- Understand, support, and apply Agile Architecture

**LEARN MORE**


This page intentionally left blank
It's said that a wise person learns from his mistakes. A wiser one learns from others' mistakes. But the wisest person of all learns from others' successes.

—Zen proverb adapted by John C. Maxwell

Abstract

A Community of Practice (CoP) is an informal group of team members and other experts, acting within the context of a program or enterprise, that has a mission of sharing practical knowledge in one or more relevant domains. CoPs are not new, nor are they mandated by Agile development. However, the Lean approach to aligning around Value Streams optimizes for delivery of value, which is a good thing. Over time, this typically causes the Lean Enterprise to pivot from a vertical, functional organization to a more flexible, horizontal line-of-business organization that can deliver value more rapidly. Further, within value streams, SAFe promotes long-lived Agile Release Trains (ARTs), which are built of people allocated to them for an extended period. What happens when practitioners of a discipline (whether or not their organization has become horizontal), who are from different programs but often have the same reporting structure, meet regularly, are led by managers and experts from their domain, and advance their specialist skills? Enter the SAFe CoP (Guild, in Spotify terminology [1]).

Details

Lean-Agile promotes cross-functional teams and programs that facilitate value delivery in the Enterprise. Similarly, Lean thinking emphasizes organizing people with different skills around a Value Stream. However, developers need to talk with other developers within or outside of the team context, testers need to talk with other testers, Product Owners need to communicate with their peers from other Agile Teams, and so on. This is critical for leveraging the multiple experiences and different types of practical knowledge available from different people at scale. That is what drives craftsmanship and persistent knowledge acquisition and facilitates the adoption of new methods and techniques.

Such inter-team communication is often supported by Communities of Practice (CoP)—informal working groups designed specifically for efficient knowledge-sharing and exploration across teams and groups of professionals, as shown in Figure 1.
Organizing a Community of Practice

A Lean enterprise has the ability to identify the relevant domains where communities of practice would be beneficial and then to foster and support such communities once they are in place.

CoPs can be ad hoc and need driven. They may or may not be permanent; they may form and disband based on current need and context. For example, an automated testing CoP could be composed of test engineers and developers who are interested in advancing these skills. An architecture and design CoP would foster the adoption of practices such as emergent design, intentional system architecture, Continuous Integration, and refactoring. It could also support the effort put into building and maintaining the Architectural Runway, foster designing for testability and deployability, deprecate old platforms, and more. Still others may be formed around Agile coaching, continuous integration, continuous delivery, coding standards, and other new practices and processes. Similarly, Scrum Masters from different Agile Teams may form a CoP to exchange facilitation best practices and experiences in building highly productive Agile Teams.

Operating a Community of Practice

A CoP is defined by the knowledge specialization of its members. Each typically has a specific learning objective, Roadmap, and backlog. Membership is fluid and changes as members take on different roles, as new needs arise, or as individual members gain the knowledge they need. CoPs may be fostered or initiated spontaneously. They are largely self-organizing, although a leader or Scrum Master equivalent (the Guild Coordinator in Spotify terminology) may organize the initiative and help maintain its momentum. CoPs meet regularly for knowledge-exchange sessions and maintain and evolve internal community websites and wikis to institutionalize their knowledge. The CoP exists only for so long as the members believe they have something to learn or contribute.

However, CoPs are created for the purpose of learning and exchanging experiences, not for coordinating dependencies or current tasks.
For instance, the Scrum Master CoP would foster learning new facilitation techniques, while actual coordination and dependency management for current work in process would happen among the same people during the Scrum of Scrums.

The Innovation and Planning Iteration presents a great time for CoPs to hold learning sessions, formal or informal, as well as other activities such as coding dojos, coaching clinics, and the like.

It is the role of Lean-Agile Leaders to encourage and support people’s desire to improve as this both helps the enterprise and builds the intrinsic motivation of knowledge workers, as is evident in SAFe Principle #9—Decentralize decision-making.

---

**LEARN MORE**

This page intentionally left blank
SAFe Core Values

Find people who share your values, and you’ll conquer the world together.
—John Ratzenberger

Abstract

Core Values are the fundamental beliefs of a person or organization. The core values are the guiding principles that dictate behavior and action. Core values can help people to know what is right from wrong; where to put their focus and help companies to determine if they are on the right path and fulfilling their business goals; and they create an unwavering and unchanging guide.

A Lean-Agile Mindset, Lean-Agile Leaders, SAFe Principles, and the extensive benefits that Lean-Agile development provides all play important roles in defining what makes SAFe safe. But in synthesis, there are four Core Values that SAFe honors, supports, and helps deliver: Alignment, Built-in Quality, Transparency, and Program Execution. If an Enterprise does those four things well, a lot of goodness will surely follow.

Details

SAFe is broad and deep and is based on both Lean and Agile principles. That’s what it’s built on, but what does SAFe itself stand for? SAFe upholds four Core Values: Alignment, Built-in Quality, Transparency, and Program Execution.

These are illustrated in Figure 1, and each is discussed in the paragraphs that follow.
Alignment

Like cars out of alignment, misaligned companies can develop serious problems. They are hard to steer and they don’t respond well to changes in direction [1]. Even if it’s clear where everyone thinks they’re headed, the vehicle is unlikely to get them there.

Alignment scales. It is a necessary condition to be able to address the business reality of fast-paced change, turbulent competitive forces, and geographically distributed teams. While empowered Agile Teams are good (even great), the responsibility for strategy and alignment cannot rest with the accumulated opinions of the teams, no matter how good they are. Rather, alignment must be based on the Enterprise business objectives. Here are some of the ways in which SAFe supports alignment:

- It starts at the strategy level of the portfolio, is reflected in Strategic Themes and the Portfolio Backlog, and then moves down through the Vision, Roadmap, and Program Backlogs to the Team Backlogs. All is visible. All is debated. All is resolved. All is known.
- It is supported by clear lines of content authority, starting at the portfolio and then resting primarily with the Product and Solution Management roles, and extending to the Product Owner role
- PI Objectives and Iteration Goals are used to communicate expectations and commitments
- Cadence and synchronization are applied to ensure that things stay in alignment, or that they drift only within reasonable economic and time boundaries

Figure 1. SAFe core values: alignment, built-in quality, transparency, program execution
• Program architecture, User Experience guidance, and governance help ensure that the Solution is technologically sound, robust, and scalable

• Lean prioritization keeps the stakeholders engaged in continuous, agreed-to, rolling-wave prioritization, based on the then-current context and changing fact patterns

Alignment, however, does not imply or encourage command and control. Instead, it provides a foundation for the enterprise where business objectives and outcomes are the continued focus. It also encourages decentralized technical and economic decision-making, thereby enabling those who implement value to make better local decisions.

**Built-in Quality**

Built-in Quality ensures that every increment of the solution reflects quality standards. Quality is not “added later.” Built-in quality is a prerequisite of Lean and flow; without it, the organization will likely operate with large batches of unverified, unvalidated work. Excessive rework and slower velocities are the likely outcome. There can be no ambiguity about the importance of built-in quality in large-scale systems. It is mandatory.

**Software**

In complex solutions, software functionality often represents a fast-changing and increasingly high-investment area. In addition, given the high levels of complexity and the manual nature of much of the work, it is often the source of many solution defects. The relatively lower cost of change encourages rapid adaptation, which is good. But if attention is not paid, the software design may quickly erode, negatively affecting quality and velocity.

Put simply, *you can’t scale crappy code*. The Agile Manifesto certainly focused on quality: “Continuous attention to technical excellence and good design enhances agility” [2]. To address software quality in the face of rapid change, software practitioners have developed and evolved a number of effective practices, many of which are largely inspired by eXtreme Programming. These include:

• Test-First: Test-Driven Development (TDD), Acceptance Test-Driven Development (ATDD), and Behavior-Driven Development (BDD)

• Continuous Integration

• Refactoring

• Pair work

• Collective ownership
**Hardware**
But coding aside, no one can scale crappy components or systems, either. Hardware elements—electronics, electrical, fluidics, optics, mechanical, packaging, thermal, and many more—are a lot less “soft.” Errors here can introduce a much higher cost of change and rework. Tips to avoid this include:

- Frequent design cycles and integration [3]
- Collaborative design practices
- Model-Based Systems Engineering
- Set-Based Design
- Investment in development and test infrastructure

**System Integration**
Eventually, different components and subsystems—software, firmware, hardware, and everything else—must collaborate to provide effective solution-level behaviors. Practices that support solution-level quality include:

- Frequent system and solution-level integration
- Solution-level testing of functional and Nonfunctional Requirements
- System and Solution Demos

**Transparency**
Solution development is hard. Things go wrong or do not work out as planned. Without transparency, facts are obscure and hard to come by. This results in decisions based on speculative assumptions and lack of data. No one can fix a secret.

For that **trust** is needed, because without trust no one can build high-performing teams and programs, or build (or rebuild) the confidence needed to make and meet reasonable commitments. Trust exists when one party can confidently rely on another to act with integrity, particularly in times of difficulty. And without trust, working environments are a lot less fun and motivating.

Building trust takes time. Transparency is the enabler for trust. SAFe helps an enterprise achieve transparency:

- Executives, Portfolio Managers, and other stakeholders are able to see into the Portfolio Kanbans and program backlogs, and they have a clear understanding of the PI goals for each train
- Programs have visibility into the team’s backlogs, as well other program backlogs
- Teams and programs commit to short-term, clear, and visible commitments. They routinely meet them.
• Programs Inspect and Adapt with all relevant stakeholders; lessons learned are incorporated.

• Teams and programs have visibility into business and architecture Epic Kanban systems. They can see what might be headed their way.

• Status reporting is based on objective measures of working systems

• Everyone can understand the velocity and WIP of the teams and programs; strategy and the ability to execute are aligned

**Program Execution**

Of course, none of the rest of SAFe matters if teams can’t execute and continuously deliver value. Therefore, SAFe places an intense focus on working systems and resultant business outcomes. This isn’t only for the obvious reasons. History shows us that while many enterprises start the transformation with Agile Teams, they often become frustrated as even those teams struggle to deliver larger amounts of solution value reliably and efficiently.

That is the purpose of the Agile Release Train, and that is why SAFe focuses implementation initially at the Program Level. In turn, the ability of Value Streams to deliver value depends on the ability of the ARTs.

But with **alignment, transparency, and built-in quality** on the team’s side, they have a little “wind at their back.” That enables a focus on **execution**. And if they struggle—and they will, because complex solution development is **hard**—they have the cornerstone of the inspect and adapt workshops. In that way, they close the loop and execute better and better during each Program Increment.

But program execution can’t just be a team-based, bottom-up thing. Successful Lean-Agile execution at scale requires not just the teams but the active support of their Lean-Agile Leaders, who couple their internal leadership with an orientation toward system and Customer outcomes. That creates a persistent and meaningful context for the teams and their stakeholders.

That’s the way the successful teams and programs are doing it, and that’s why they are getting the many benefits—employee engagement, productivity, quality, and time to market—that Lean-Agile enterprises so enjoy.

---

**LEARN MORE**


This page intentionally left blank
It all starts with a Lean-Agile Mindset.
—SAFe Authors

Abstract
SAFe is based on a number of newer paradigms in modern systems and software engineering, including Lean and systems thinking, product development flow, and Agile development. As reflected at the Team Level, Agile provides the tools needed to empower and engage teams to achieve unprecedented levels of productivity, quality, and engagement. But a broader and deeper Lean-Agile Mindset is needed to support Lean and Agile development at scale across the entire Enterprise.

Thinking Lean
Much of the thinking in Lean is represented in the SAFe “House of Lean” icon. It is organized around six key constructs. The “roof” represents the goal of delivering Value, the “pillars” support that goal via Respect for People and Culture, Flow, Innovation, and Relentless Improvement. Lean-Agile Leadership provides the foundation on which everything else stands.

Embracing Agility
In addition, SAFe is built entirely on the skills, aptitude, and capabilities of Agile Teams and their leaders. And while there is no one definition of what an Agile method is, the Agile Manifesto provides a unified value system that has helped inaugurate Agile methods into mainstream development.

Together, these create the Lean-Agile Mindset, part of a new management approach and an enhanced culture, one that provides the leadership needed to drive a successful transformation, and one that helps both individuals and businesses achieve their goals.

Details
The SAFe House of Lean
While initially derived from Lean manufacturing [1], the principles and practices of Lean thinking as applied to software, product, and systems development are now deep and extensive.
For example, Ward [2], Reinertsen [3], Poppendieck [4], Leffingwell [5], and others have described aspects of Lean thinking that put many of the core principles and practices into a product development context. In combination of these factors, we present the SAFe House of Lean, as illustrated in Figure 1, which is inspired by “houses” of Lean from Toyota and others.

**The Goal – Value**

The goal of Lean is inarguable: to deliver the **maximum Customer value in the sustainably shortest lead time**, while providing the highest possible quality to Customers and society as a whole. High morale, safety, and Customer delight are further tangible targets and benefits.

**Pillar 1 – Respect for People and Culture**

SAFe is a systematic framework for implementing Lean-Agile development at scale, but it does not instantiate itself, nor does it perform any real work. **People do all the work.** Respect for people and culture is a fundamental value of the SAFe House of Lean. People are empowered to evolve their own practices and improvements.
Management challenges people to change and may even indicate what to improve, but the teams and individuals learn problem-solving and reflection skills, and they make the appropriate improvements.

Culture is the driving force behind this behavior. To evolve a truly Lean organization, the culture will need to change. In order for that to happen, the organization and its leaders must change first. And culture and people are not solely an internal construct. The culture of the organization extends to long-term relationships with Suppliers, partners, Customers, and the broader community that supports the Enterprise.

Where there is urgency for positive change, improvements in culture can be achieved gradually by, first, understanding SAFe values and principles; second, implementing SAFe practices; and third, delivering positive results. Changes to culture will follow naturally.

**Pillar 2 – Flow**

The key to successful execution in SAFe is establishing a continuous flow of work that supports incremental value delivery, based on continuous feedback and adjustment. Establishing continuous flow is critical to fast value delivery; effective quality practices; continuous improvement; and effective, evidence-based governance. The principles of flow, reflected in this pillar of the House of Lean, constitute an important subset of the SAFe Lean-Agile Principles and are instantiated in various practices throughout. These include understanding the full Value Stream, visualizing and limiting WIP, reducing batch sizes and managing queue lengths, and prioritizing work based on the cost of delay. Lean also has a primary focus on Built-in Quality, fast feedback, and the identification and constant reduction of delays and non-value-added activities.

These constructs provide a pivotal change to a better understanding of the system development process and provide new thinking, tools, and techniques that leaders and teams can use to move from phase-gated processes to more continuous value delivery.

**Pillar 3 – Innovation**

Flow builds a solid foundation for the delivery of value, but without innovation, both product and process will stagnate. Innovation is a critical part of the SAFe House of Lean. In support of innovation, Lean-Agile Leaders:

- “Get out of the office” and into the actual workplace where value is produced and products are created and used (gemba). As Taiichi Ohno put it, “No useful improvement was ever invented at a desk.”

- Provide time and space for people to be creative. Time for innovation must be purposeful. Innovations can rarely occur in the presence of 100% utilization and continuous firefighting. SAFe’s Innovation and Planning Iteration is one such opportunity.
• Apply innovation accounting [6]. Establish non-financial, non-vanity Metrics that provide fast feedback on the important elements of the new innovation.

• Validate the innovation with Customers, then *pivot without mercy or guilt* when the hypothesis needs to change

### Pillar 4 – Relentless Improvement

The fourth pillar is relentless improvement. With this pillar, the organization is guided to become a learning organization through continuous reflection and relentless improvement. A constant sense of competitive danger drives the learning organization to aggressively pursue improvement opportunities. Leaders and teams do the following systematically:

- Optimize the whole, not the parts, of both the organization and the development process
- Consider facts carefully, then act quickly
- Apply Lean tools and techniques to determine the root cause of inefficiencies and apply effective countermeasures quickly
- Reflect at key Milestones to openly identify and address the shortcomings of the process at all levels

### Foundation – Leadership

The foundation of Lean is leadership, which is the ultimate enabling force for team success. Here, SAFe’s philosophy is simple: *The ultimate responsibility for adoption and success of the Lean-Agile paradigm lies with the enterprise’s existing managers, leaders, and executives.* “Such a responsibility cannot be delegated” (Deming [7]) to Lean/Agile champions, Lean/Agile working groups, development teams, a PMO, process teams, outside consultants, or any other party. To achieve success, leaders must be trained in these new and innovative ways of thinking and exhibit the principles and behaviors of Lean-Agile leadership.

Lean thinking deviates from common experience with Agile, which was often introduced as a team-based process that tended to exclude management. That does not scale. Here is a key differentiator between traditional Agile and one of the key drivers for SAFe:

In traditional Agile, the expectation has been that management *simply supports* the teams and helps eliminate impediments as they arise. In Lean-Agile development, the expectation is that management *leads* the teams, embraces the values of Lean, is competent in the basic practices, proactively eliminates impediments, and takes an active role in driving organizational change and facilitating relentless improvement.
The Agile Manifesto

In the 1990s, in response to the many challenges of waterfall development methods, a number of lighter-weight and more iterative development methods arose. In 2001, many of the leaders of these methods came together in Snowbird, Utah. While there were differences of opinion on the specific merits of one method over another, the attendees agreed that their common values and beliefs dwarfed the differences in approach. The result was a *Manifesto for Agile Software Development* [8], which was a turning point that helped unify the approach and started to bring the benefits of these innovative methods to the industry at large. The Manifesto consists of a value statement, as exhibited in Figure 2, and a set of principles, as exhibited in Figure 3.

![The Values of the Agile Manifesto](agilemanifesto.org)

*Figure 2. Values of the Agile Manifesto*
The Principles of the Agile Manifesto

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity—the art of maximizing the amount of work not done—is essential.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Along with the various Agile methods, the Manifesto provides the Agile foundation for effective, empowered, self-organizing teams. SAFe extends this foundation to the level of teams of teams and applies Lean thinking to understand and relentlessly improve the systems that support the teams in their critical work.
LEARN MORE


This page intentionally left blank
SAFe Principles

The impression that “our problems are different” is a common disease that afflicts management the world over. They are different, to be sure, but the principles that will help to improve the quality of product and service are universal in nature.

—W. Edwards Deming

SAFe is based on a number of immutable, underlying Lean and Agile principles. These are the fundamental tenets, the basic truths and economic underpinnings that drive the roles and practices that make SAFe effective. The nine principles are:

#1-Take an economic view
#2-Apply systems thinking
#3-Assume variability; preserve options
#4-Build incrementally with fast, integrated learning cycles
#5-Base milestones on objective evaluation of working systems
#6-Visualize and limit WIP, reduce batch sizes, and manage queue lengths
#7-Apply cadence, synchronize with cross-domain planning
#8-Unlock the intrinsic motivation of knowledge workers
#9-Decentralize decision-making
Why the Focus on Principles?

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

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

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

SAFe’s goal is to synthesize some of this body of knowledge and the lessons learned from hundreds of deployments into a single framework—a system of integrated, proven practices that has been demonstrated to bring substantial improvements in employee engagement, time to market, solution quality, and team productivity. However, given the complexity of the industry challenges already discussed, there is truly no off-the-shelf solution to the unique challenges every enterprise faces. This means that some tailoring and customization may be required, as not every SAFe-recommended practice will apply equally well in every circumstance. Therefore, we always endeavor to make certain that SAFe practices are grounded in fundamental, and reasonably immutable, principles. In that way, we can be confident that they apply well in the general case. And when and if they don’t, the underlying principles can guide those doing the implementation to make sure that they are moving on a continuous path to the “shortest sustainable lead time, with best quality and value to people and society.” There is value in that too.

The nine SAFe Principles are discussed in greater detail in the next chapter.
Implementing 1-2-3

Implementing 1-2-3

It is not enough that management commit themselves to quality and productivity, they must know what it is they must do. Such a responsibility cannot be delegated.

—W. Edwards Deming

Abstract

Implementing the changes necessary to become a Lean-Agile technology enterprise is a substantial change for most organizations. Embracing a Lean-Agile Mindset, understanding and applying the Lean-Agile principles, and effectively implementing the SAFe practices all come before the business benefits. And, of course, the culture must evolve, too.

While SAFe is a freely revealed body of knowledge, available to all, it does not implement itself, nor does it prescribe the organizational change management process that is typically required for successful implementation. We leave that to the enterprise, because only they know their specific context, and they—typically assisted by their partners—must own the transformation.

But many enterprises have gone down this path already (see the “Case Studies” articles online at www.scaledagileframework.com), and the lessons learned are now becoming more widely accessible. Based on the learnings from hundreds of SAFe implementations, Scaled Agile, Inc., the owner of SAFe, has developed a basic Implementing SAFe 1-2-3 pattern for successful SAFe adoption. It provides a simple roadmap that helps gets everyone aligned to a common implementation strategy.

This article describes an overview of this successful pattern for SAFe implementation, along with pointers to the growing community of service providers who are ready and willing to help your enterprise make this critical transformation.

Details

Figure 1 on the next page provides a high-level summary of the Implementing SAFe 1-2-3 approach. Each of the numbered items in this strategy is described in the paragraphs that follow.
1. Train Implementers and Lean-Agile Change Agents

Given the scope, challenge, and impact of rollouts, successful adoption of SAFe requires most enterprises to use a combination of internal and external change agents, leaders, mentors, and coaches. These people need to be skilled in teaching and delivering SAFe. To achieve this, Scaled Agile, Inc. provides an *Implementing SAFe 4.0 with SPC Certification* program. After taking this class, attendees will be able to:

- Lead an enterprise Agile transformation with SAFe
- Implement SAFe
- Launch Agile Release Trains and proctor and continuously improve the trains via Inspect and Adapt workshops

Those who take and pass the optional SPC Certification exam (included) will be licensed to:

- Train managers and executives in Leading SAFe and act as a SAFe Agilist (SA) certifying agent
- Train practitioners in SAFe 4.0 for Teams and act as a SAFe Practitioner (SP) certifying agent

---

*Figure 1. Implementing SAFe 1-2-3*
The Implementing SAFe 4.0 with SAFe Program Consultant (SPC4) Certification course is delivered by certified SPC Trainers (SPCT) in open enrollment or on-site settings worldwide. Service providers who specialize in SAFe and Scaled Agile Partners can be found online at www.scaledagile.com. There are many independent SPCs as well; these can be found at the Scaled Agile SPC membership site at www.scaledagileacademy.com.

2. Train all Executives, Managers, and Leaders

It is critical that executives, managers, and leaders understand what is required to lead a Lean-Agile transformation, including how and why SAFe works. To help achieve this, Scaled Agile, Inc., provides a two-day course, Leading SAFe 4.0, Leading the Lean-Agile Enterprise with the Scaled Agile Framework. After attending, participants will be able to:

- Adopt a Lean-Agile mindset
- Apply Lean and Agile principles; base daily decisions on this long-term philosophy; understand, exhibit, and teach these principles
- Understand the practices, roles, activities, and artifacts of the Scaled Agile Framework
- Unlock the intrinsic motivation of knowledge workers
- Learn the practices and tools of relentless improvement and teach employees problem-solving and corrective-action skills
- Become hands-on in the new process adoption, eliminate impediments, and facilitate organizational change management
- Take responsibility for Lean-Agile implementation success

The audience for this class is executives, managers, and change agents responsible for leading a Lean-Agile change initiative, whereby they gain the knowledge necessary to lead the SAFe adoption.

A certification exam is optional for this course. Those who pass the optional exam will be certified as SAFe Agilist (SA), and will receive one year’s membership to that community and its benefits.

The Leading SAFe 4.0 course is delivered by certified SPC consultant/trainers in open enrollment or on-site settings worldwide. Service providers include Scaled Agile, Inc.; Scaled Agile Partners; and independent SPCs.

3. Train Teams and Launch Agile Release Trains

The primary value delivery mechanism in the enterprise is the Agile Release Train, but starting these trains is not a trivial task. One proven starting mechanism is an Agile Release Train Quickstart.
Suitable after some significant up-front preparation, the QuickStart is a one-week training and immersion program that:

- Organizes 50 – 100 team members into Agile Teams, training them simultaneously in the principles of Lean, Agile, and SAFe
- Aligns the teams on the train to a common mission and spends two days in face-to-face support of planning the next Program Increment
- Introduces prospective Product Owners and Scrum Masters to the skills and activities unique to their roles in the new Agile enterprise
- Builds context and a cadence-based, rolling-wave planning and delivery model that continuously incorporates business objective setting and program commitments, effective and reliable program execution, and adaptive feedback

SPCs can provide these services and download and use the SAFe ART Launch Pack (member login required) to prepare for a successful launch. It contains the tools to prepare the organization, programs, teams, and individuals for success and continuous improvement. You may also want to consider licensing the ART Training and Launch Pack Bundle. This bundle provides both the courseware and tools needed to quickly and effectively launch Agile Release Trains.

Supporting Consulting Activities

Once the enterprise has a critical mass of in-house Lean-Agile Leaders, and a few Agile Release Trains rolling, a variety of consulting activities may be applicable and beneficial. These could include coaching the train, training specialist roles, and continuous improvement.

Coaching the Train

By sharing their knowledge and experience, coaches can help teams and individuals improve their newfound skills by:

- Providing program consulting and team coaching to build the organization’s Lean-Agile capabilities
- Facilitating Agile Release Train readiness, including Program Backlog refinement and more
- Facilitating inspect and adapt workshops
- Facilitating Portfolio planning workshops
- Implementing relevant Metrics and governance
- Mentoring executives, managers, and other program stakeholders in SAFe adoption
- Shadowing and mentoring Release Train Engineers
Many of these activities are supported by various Scaled Agile workshop kits, which are available to SPCs in good standing.

Training Specialist Roles
It is important to train specialists—including prospective Product Owners and Scrum Masters—in the principles and practices unique to their roles. Training courses for this purpose include:

- **SAFe 4.0 Scrum Master Orientation** – Half- to 1-day orientation to the role of a SAFe Scrum Master
- **SAFe 4.0 Product Manager / Product Owner with PMPO certification** – This 2-day certification course is for Product Managers, Business Owners, and Product Owners who will learn how to manage and prioritize backlogs, participate in SAFe events, define and support epics, capabilities, features, and user stories, and manage stakeholders at the various levels of the enterprise.
- **SAFe 4.0 Advanced Scrum Master with ASM Certification** – This 2-day advanced, certification course prepares current Scrum Masters for their leadership role in facilitating Agile team, program, and enterprise success. It enhances the Scrum paradigm with an introduction to scalable engineering and DevOps practices; the application of Kanban to facilitate flow; supporting interactions with architects, product management, and other critical stakeholders; and tips and techniques for building high-performing Agile teams.

Note: The courseware offerings are always advancing, so be sure and check ScaledAgile.com for the latest updates.

Continuous Improvement
Once the transformation is under way, there are a variety of opportunities for sustaining and enhancing improvements in speed and quality that can be best facilitated by the extensive community of skilled professionals. These activities can include Agile Release Train health checks; Portfolio, Value Stream, Agile Release Train, and Team agility self-assessments; and facilitated Inspect and Adapt sessions. For help, we again refer you to your in-house SPCs, Scaled Agile Partners, and other independent SPCs.

Guidance and Governance with Enterprise SAFe
For those who would benefit from being able to modify a custom version of the Scaled Agile Framework website, all SAFe content is available for enterprise licensing. Organizations that are scaling Lean-Agile best practices leverage Enterprise SAFe so they can have access to the most up-to-date content for their teams and the thought leaders at Scaled Agile, Inc.

Enterprise SAFe allows organizations to align around common process objectives while providing the ability to adapt SAFe to their unique needs and culture. Enterprise SAFe allows organizations to create a custom version of the Scaled Agile Framework website while maintaining automated updates as the methodology advances.
Fully Adaptable to Your Organization’s Context
Provisioned by Scaled Agile, Inc., and built on WordPress, Enterprise SAFe supports adaptation that enables organizations to revise the graphical representation of the SAFe Big Picture as well as the entire SAFe content offering. Content is controlled locally by the enterprise via a set of tools that support accepting or rejecting framework updates from Scaled Agile, Inc., as well as adding custom content, such as articles, icons, labels, and graphics.

Enterprise SAFe features include:

- A WordPress publishing platform that allows you to start capturing the specifics of your custom SAFe implementation in a matter of minutes
- A customizable Big Picture in Adobe Illustrator that allows you to capture key modifications to the framework at the front end of the website
- A PowerPoint version of all SAFe artwork that allows you to change the graphics that are integral to the story
- Provisions to modify or extend SAFe with your custom process content
- Local control of custom content via a set of tools that allow reviewing differences between local pages and content updates, and accepting or rejecting updates
- Administration utility for easy management of large numbers of user accounts

Enterprise SAFe is provisioned by Scaled Agile, Inc., via a private and secure cloud-based website.
Index

3Cs: Card, Conversation, Confirmation, 136–137
3-level SAFe, Big Picture, 1
4-level SAFe, Big Picture, 1
80/20 rule, 256–257

A


Acceptance tests
  automated, 502
  system-level, 498–499
  template/checklist, 502

Adaptive planning, 368

Agenda for
  iteration planning meeting, 116
  iteration retrospective, 131
  PI planning event, 216–218
  solution demo, 381
  SoS (Scrum of Scrums), 210
  system demo, 235
  team demo, 127

Agile architecture
  abstract, 371
  architectural runway, 371
  in the Big Picture, 371–378
  definition, 511
  details, 371–378
  emergent design, 371
  implementers, training, 38
  intentional architecture, 371
  recording models in solution intent, 376
  tool for solution intent, 203
  value stream level, 371–378

Agile architecture, basic principles
  architectural collaboration, 372, 373–374
  coding alternative approaches, 375–376
  emergent design, 372–373
  fostering a culture of innovation, 377
  implementing architectural flow, 377
  intentional architecture, 373
  keeping it simple, 375
  modeling alternative approaches, 375–376
  responsibility for testing, 376–377
  system size versus runway length, 374–375

Agile contracts
  abstract, 503
  in the Big Picture, 503–509
  collaborative approach, 505–506
  details, 503–509
  firm fixed-price, 503–505
  SAFe Managed-Investment Contracts, 506–509
  with suppliers, 393
  time and materials, 503–505

Agile Development in High-Assurance and Regulated Environments, 352

Agile Manifesto, 31–32. See also Lean-Agile mindset; SAFe principles.


Agile Software Requirements, 497

Agile Teams
  abstract, 77
  in the Big Picture, 77–81
  choosing Agile practices, 79
  collaborative demos, 80
  collaborative learning, 81
  definition, 511
  details, 77–78
  estimating work, 80
  integration with other teams, 80
  intense collaboration, 78–79
  PI planning, 80
  relentless improvement, 81
  roles and responsibilities, 78
  skill requirements, 77–78
  team level, 77–81

Agile Testing, 497

Agile Testing Matrix, 497–499

Alignment
  core value of SAFe, 22–23
  definition, 22–23
  development manager, 14
  scaling, 22

All-at-once approach to NFRs, 204

Analysis stage, portfolio Kanban, 445, 446

Architectural collaboration, principle of Agile architecture, 372, 373–374

Architectural flow, implementing, 377

Architectural runway
  abstract, 265
  Agile architecture, 371
  in the Big Picture, 265–270
creating, 267–269
definition, 511
destruction over time, 269
details, 265–270
emergent design, 265–266
extending, 266–267, 270
intentional architecture, 266
potential problems, 266
program level, 265–270
source of the metaphor, 270
system size versus runway length, 374–375
Architecture enablers, 245, 246
ARTs (Agile Release Trains)
abstract, 159
in the Big Picture, 159–165
Business Owners, 161
CI (continuous integration), 492
goal, 40
component teams, 164–165
definition, 2, 511
details, 159–165
DevOps, 161
feature teams, 164–165
key concepts, 160
key roles, 161
launching, 39–40
local versus global optimization, 160
mixing with traditional methods, 158
operating principles, 160–161
PM (Product Management), 161
program level, 159–165
RTE (Release Train Engineer), 161
Self Assessment, 318–319
self-managing, 433
Shared Services, 161
steering. See RTE (Release Train Engineer); VSE
(Value Stream Engineer).
System Architects/Engineers, 161
System Team, 161
teams on the train, 164–165
UX designers, 161
ARTs (Agile Release Trains), for value streams
coordinating, 481
multi-ART value streams, 480
multiple value stream ARTs, 478–479
single value stream ARTs, 479–480
ARTs (Agile Release Trains), organizing
around capabilities, 163–164
around subsystems, 163–164
effective train size, 162
factors to consider, 162–165
teams on the train, 164–165
value stream size, 162–163
value streams, splitting, 163–165
Assigning business value, Business Owners, 223

Assuming variability (SAFe Principle #3)
in the Big Picture, 55–56
overview, 55–56
solution intent, 353
ATDD (Acceptance Test-Driven Development). See also
Test-first methods.
in the Agile Testing Matrix, 500
definition, 149
PO (Product Owner) support for, 84
test-first methods, 500
Autonomy, motivating knowledge workers
(SAFE Principle #8), 68
Average lead time, Kanban, 101–102

B
Backlog refinement. See also Team backlog.
PO (Product Owner), 84
program backlogs, 194
role of PO (Product Owner), 84
team backlog, 107–108
value stream backlogs, 194
value stream Kanban, 191
Backlogs
creating improvement backlog items, 257–258
Little’s Law, 197
NFRs as backlog constraints, 201–202
portfolio level, 449–452
team level, 105–108
Backlogs, portfolio level
in the Big Picture, 449–452
definition, 516
Backlogs, program level
abstract, 193
in the Big Picture, 193–197
capacity allocation, 195–196
definition, 516
details, 193–197
Little’s Law, 197
PI planning, 195
prioritizing, 194
program backlogs, 193–197
queues, 197
refining, 194
solution integrity, optimizing, 195–196
value, optimizing, 195–196
value stream backlogs, 193–197
wait times, 197
Backlogs, team level
abstract, 105
in the Big Picture, 105–108
capacity allocation, 107
definition, 520
details, 105–108
refinement, 107–108
specification workshops, 108
system health, optimizing, 107
value delivery, optimizing, 107

Backlogs, value stream level
abstract, 193
in the Big Picture, 193–197
capacity allocation, 195–196
definition, 516, 521
details, 193–197
Little's Law, 197
PI planning, 195
pre- and post-PI planning meetings, 386
prioritizing, 194
queues, 197
refining, 194
solution integrity, optimizing, 195–196
value, optimizing, 195–196
wait times, 197

Batch sizes, reducing. See Reduce batch sizes
(SAFe Principle #6).

Beck, Kent, 375

Beyond Entrepreneurship, 420

Beyond project cost accounting, 454–461

Big Picture, guidance
   Agile contracts, 503–509
   CI (continuous integration), 491–496
test-first methods, 497–502
web address, 489

Big Picture, portfolio level
   budgets, 453–461
   CapEx (capital expense), 463–471
team level, 3
enterprise, 417–422
Enterprise Architect, 439–442
Epic Owners, 435–437
epics, 483–488
illustration, 7
introduction, 413–415
OpEx (operational expense), 463–471
portfolio backlog, 449–452
portfolio Kanban, 443–447
PPM (Program Portfolio Management), 429–434
portfolio level, 7
value streams, 473–482
web address, 411

Big Picture, program level
   architectural runway, 265–270
   ART (Agile Release Train), 159–165
   Business Owners, 221–224
   capabilities, 237–242
develop on cadence, release any time, 259–264
enablers, 243–247
features, 237–242
I&A (inspect and adapt), 253–258
illustration, 4

innovation and planning iteration, 249–252
introduction, 155–158
NFRs (nonfunctional requirements), 199–206
PI (Program Increment), 207–212
PI objectives, 225–231
PI planning, 213–220
Product Management, 177–181
program backlogs, 193–197
program Kanban, 187–191
RTE (Release Train Engineer), 167–170
Solution Architect/Engineering, 171–175
Solution Management, 177–181
System Architect/Engineering, 171–175
system demo, 233–236
value stream backlogs, 193–197
value stream Kanban, 187–191
VSE (Value Stream Engineer), 167–170
web address, 153
WSIF (Weighted Shortest Job First), 183–186

Big Picture, SAFe foundation
   3-level SAFe, 1
   4-level SAFe, 1
   CoPs (Communities of Practice), 17–19
core values, 21–25
definition, 1
House of Lean, 27, 28–30
Implementing 1-2-3, 37–42
Lean-Agile Leaders, 11–15
Lean-Agile mindset, 27–32
portfolio level, 7
program level, 4
spanning palette, 5
system demo, 233–236
web address, 411

Big Picture, SAFe principles
   assuming variability (Principle #3), 55–56
cadence (Principle #7), 63–65
cross-domain planning (Principle #7), 63–65
decentralizing decision-making (Principle #9), 71–72
economic view (Principle #1), 45–49
incremental build (Principle #4), 57–58
integrated learning cycles (Principle #4), 57–58
management queue lengths (Principle #6), 61–62
motivating knowledge workers (Principle #8), 67–69
objective milestones (Principle #5), 59–60
preserving options (Principle #3), 55–56
reduce batch sizes (Principle #6), 61–62
synchronization (Principle #7), 63–65
strengths thinking (Principle #2), 51–53
visualize and limit WIP (Principle #6), 61–62
web address, 43

Big Picture, spanning palette
   DevOps, 273–278
   illustration, 5
metrics, 307–321
milestones, 323–329
Release Management, 283–285
releases, 331–336
roadmaps, 301–306
Shared Services, 287–289
System Team, 279–282
UX (User Experience), 291–294
Vision, 295–300
web address, 271
Big Picture, team level
Agile Teams, 77–81
built-in quality, 147–152
demos, 125–127
introduction, 75–76
iteration execution, 119–124
iteration goals, 143–145
iteration planning, 113–117
iteration retrospective, 129–131
iterations, 109–111
Product Owners, 83–87
Scrum Master, 89–91
ScrumXP teams, 93–97
solution demo, 125
stories, 133–141
system demo, 125
team backlog, 105–108
team demo, 125–127
team Kanban, 99–104
web address, 73
Big Picture, value stream level
abstract, 339
Agile architecture, 371–378
Customers, 395–399
details, 339–343
economic framework, 347–349
introduction, 339–343
MBSE (Model-Based Systems Engineering), 359–364
set-based design, 365–369
solution context, 405–409
solution demo, 379–382
solution intent, 351–358
solutions, 401–404
suppliers, 389–394
value stream coordination, 343–346
web address, 337
Big visual information radiators (BVIRs), 95
Books and publications
Agile Development in High-Assurance and Regulated Environments, 352
Agile Software Requirements, 497
Agile Testing, 497
Beyond Entrepreneurship, 420
Beyond Project Cost Accounting, 414
The First Decision Rule Principle, 49. See also
Decentralizing decision-making (SAFe Principle #9).
The Lean Startup, 420
"Mixing Agile and Waterfall Development," 158, 391
The Principle of Continuous Economic Trade-Offs, 49
The Principle of Optimum Decision Making, 49
The Principle of Quantified Cost of Delay, 49
Release Planning Readiness Checklist, 215
The Sunk Cost Principle, 49
Technical Strategies for Agile and Waterfall Interoperability at Scale,” 391
Bounded NFRs, 203
Box, George E. P., 359
Brainstorming, I&A (inspect and adapt) workshop, 257
Budgets
definition, 511
tuition, 418, 421
Lean-Agile budgeting, 348, 432
portfolio level, 453–461
PPM (Program Portfolio Management), 432
for value streams, portfolio level, 477–478
Budgets, portfolio level
abstract, 453
approving epic level initiatives, 460
beyond project cost accounting, 454–461
in the Big Picture, 453–461
cost-center budgeting, problems with, 455–456
details, 453–461
effects of delays, 457
empowering value stream content authority, 459–460
fiscal governance with dynamic budgeting, 461
funding value streams, not projects, 458–459
Lean-Agile Budgeting: Beyond Project Cost Accounting, 454–455
problems of traditional cost accounting, 455–457
project-based constraints, 456–457
providing objective evidence of fitness for purpose, 460.
See also Objective milestones (SAFe Principle #5).
Built-in quality, core value of SAFe
abstract, 147
in the Big Picture, 147–152
business benefits, 148
CI (continuous integration), 148–149, 152
collective ownership, 150–151
definition, 23, 511
design verification, 152
details, 147–152
development manager, 14
firmware, 151–152
hardware, 24, 151–152
iteration execution, 122
MBSE (Model-Based Systems Engineering), 362
pair programming, 150
pair work, 150
refactoring, 150
scaling crappy code/components, 23–24
software, 23, 148–151
system integration, 24
        team level, 147–152
Business benefits, pre- and post-PI planning meetings, 384
Business context, pre- and post-PI planning meetings, 385
Business drivers, enterprise, 421
Business epics. See Epics.
Business objectives. See Strategic themes.
Business Owners
        abstract, 221
        ART (Agile Release Train), 161
        assigning business value, 223
        in the Big Picture, 221–224
definition, 512
details, 221–224
        at I&A (inspect and adapt) workshop, 223–224
        importance of, 224
during PI execution, 223–224
during PI planning, 222
        prior to PI planning, 222
        program level, 221–224
        roles and responsibilities, 221–224
Business value, assigning
to PI objectives, 229–230
role of Business Owners, 223
BVIRs (big visual information radiators), 95

C
Cadence (SAFe Principle #7)
in the Big Picture, 63–65
objective milestones (SAFe Principle #5), 60
overview, 63
value stream coordination, 344–345
Cadence-based integration points, 58
Calendars, sample IP iteration calendar, 251–252
Capabilities
        abstract, 237
        in the Big Picture, 237–242
definition, 237, 512
details, 237–242
enablers, 244
        program level, 237–242
        splitting into features, 241–242
        value stream, 240–241
Capacity allocation
        program backlogs, 195–196
team backlog, 107
        value stream backlogs, 195–196
Capacity ARTs, 163–164
CapEx (capital expense), definition, 512
CapEx (capital expense), portfolio level
abstract, 463–464
Agile development capitalization strategies, 467–468
    allowable labor efforts, 470–471
applying stories, 468–470
    in the Big Picture, 463–471
capitalization triggers in waterfall development, 466–467
capitalization versus expense criteria, 465–466
capturing labor effort, 469
categorization of features, 468
details, 463–471
FASB (Financial Accounting Standards Board), 465
FASB 86 guidelines, 465
portfolio level, 463–471
software classification, 465
software development costs, 465
by story count, 470
by story hours, 469
by story points, 469–470
Capitalization. See also CapEx (capital expense), portfolio level.
    Agile development capitalization strategies, 467–468
    versus expense criteria, 465–466
triggers in waterfall development, 466–467
Card aspect of stories, 136
Case Studies articles, website, 37
Categorization of features for CapEx, 468
CFD (cumulative flow diagram), 101–102, 318
CI (continuous integration)
        abstract, 491
        ART integration, 492
        in the Big Picture, 491–496
built-in quality, 148–149, 152
challenges, 494–495
common cadence, 495
        component level, 491–492
definition, 512
        deprecated tests, 494
designing for testability, 496
details, 491–496
emulated environments, 495
enabling, 495–496
feature level, 491–492
implementing a culture of, 496
infrastructure, 495
integration frequency, 494
mocks, 495
regression tests, 494
solution context synchronization, 493–494
solution integration, 492–494
stubs, 495
supplier synchronization, 493–494
supporting the System Team, 496
trade-offs, 494–495
Classes of service, 102–103
Coaching the ART (Agile Release Train), 40
Coding alternative approaches, principle of Agile architecture, 375–376
Collaborating on solution intent, 354–355
Collaborating with suppliers, 392–393
Collaboration, in solution context, 408–409
Collaborative demos, Agile Teams, 80
Collaborative learning, Agile Teams, 81
Collective ownership, built-in quality, 150–151
Collins, James, 420
Communities of Practice (CoPs). See CoPs (Communities of Practice).
Compensation, motivating knowledge workers (SAFe Principle #8), 67
Competitive environment, 421
Compliance, MBSE (Model-Based Systems Engineering), 360–361, 362
Component level, CI (continuous integration), 491–492
Component tests, 498, 500
Confidence vote, pre- and post-PI planning meetings, 387
Confirmation aspect of stories, 137
Constant communication, iteration execution, 121
Consulting activities, 40–41
Content authority, PO (Product Owner), 85–86
Content management
  a Lean-Agile approach, 177–178
  at the program level, 156
  value stream coordination, 345–346
Content readiness, PI planning event, 215
Continuous delivery, 46–47, 262
Continuous improvement. See Kanban; Relentless improvement.
Continuous integration (CI). See CI (continuous integration).
Continuous learning, innovation and planning iteration, 251
Continuous management information, iteration goals, 146
Continuous value flow, PPM (Program Portfolio Management), 432
Contracts, Agile. See Agile contracts.
Conversation aspect of stories, 136–137
Coordination. See Value stream coordination.
CoPs (Communities of Practice)
  abstract, 17
  Big Picture, 17–19
decentralizing decision-making (SAFe Principle #9), 19
definition, 2, 512
details, 17–19
Guild Coordinator. See SM (Scrum Master). horizontal line-of-business organization, 17
operating, 18–19
organizing, 18
Scrum Master, 18–19
Core values, enterprise, 421
Core values of SAFe
  abstract, 21
  alignment, 22–23
  in the Big Picture, 22
definition, 21
details, 21–25
  list of, 2, 21. See also specific values.
  program execution, 25
  transparency, enabling trust, 24–25
Core values of SAFe, built-in quality
  abstract, 147
  in the Big Picture, 147–152
  business benefits, 148
  CI (continuous integration), 148–149, 152
  collective ownership, 150–151
definition, 23, 511
design verification, 152
details, 147–152
development manager, 14
firmware, 151–152
hardware, 24, 151–152
iteration execution, 122
pair programming, 150
pair work, 150
refactoring, 150
scaling crappy code/components, 23–24
software, 23, 148–151
system integration, 24
Cost accounting, traditional problems of, 455–457. See also Budgets.
Cost of delay
  calculating, 184–185
  primary elements of, 184
  risk reduction-opportunity enablement value, 184
  time criticality, 184
  user-business value, 184
Cost-center budgeting, problems with, 455–456
Cross-ART enablers, 246–247
Cross-domain planning, synchronization, 63–65
Cross-value stream enablers, 246–247
Culture of CI (continuous integration), implementing, 496
Culture of innovation, principle of Agile architecture, 377
Cumulative flow diagram (CFD), 101–102, 318
Customer context for solutions, 403
Customers
abstract, 395
in the Big Picture, 395–399
custom solutions, 398–399
definition, 512
details, 395–399
engagement, 397–399
general solutions, 398
internal versus external, 396–397
responsibilities, 396
value stream level, 395–399
Decentralized, rolling-wave planning, 433
Decentralizing decision-making (SAFe Principle #9)
in the Big Picture, 71–72
CoPs (Communities of Practice), 19
economic framework, 348, 349
suppliers, 391–392
System Architect/Engineering, 174
Decision-making
decentralizing. See Decentralizing decision-making
(SAFe Principle #9).
strategic decisions, 71
Decision-making framework, 71–72
Delays, effects on budgets, 457
Delivering early and often, economic view (SAFe Principle
#1), 46–47
Demand management, PPM (Program Portfolio
Management), 432
Deming, W. Edwards, 51, 393–394
Demos. See also Solution demos; System demos.
in the Big Picture, 125–127
of solutions, 403
team level, 125–127
Dependencies, managing, 145
Deployability, 276, 408–409
Deployment. See also DevOps.
automating, 277
frequency, 276
readiness, 276
value stream coordination, 346
Deployment infrastructure strategy, Enterprise Architects, 441
Deprecated tests, CI (continuous integration), 494
Design verification, built-in quality, 152
Designing for testability, CI (continuous integration), 496
Designs, solution intent, 352–353
Develop on cadence, definition, 512
Develop on cadence, release any time
abstract, 259–264
in the Big Picture, 259–264
continuous delivery, 262
details, 259–264
PI (Program Increment), 208
program level, 259–264
releases, 263–264, 332
separating development concerns from release, 262–263
synchronization, 259–261
Developing, solution intent, 354–355
Development environments, DevOps, 276
Development infrastructure, building, 280
Development infrastructure strategy, Enterprise Architects, 441
Development manager
alignment, 14
built-in quality, 14
overview, 13
personnel and team development, 13–14
program execution, 14
responsibilities, 13
transparency, 14
Development value streams, 474–475, 477, 478–480
DevOps. See also Deployment.
abstract, 273
ART (Agile Release Train), 161
automatically building environments, 277
automating the deployment process, 277
in the Big Picture, 273–278
definition, 513
deployability, 276, 277–278
deployment frequency, 276
deployment readiness, 276
details, 273–278
development and test environments, 276
importance to the value stream, 273–274
Internet of Things, 274
Murphy's Law, 275
in a production-equivalent staging environment, 275–276
recommended practices, 275–278
solution architecture, 277–278
spanning palette, 273–278
on the train, 274
typical team members, 274
version control, 277
Distinctive competence, 421–422
Distributed, governed UX development, 294
Enabler work, scope of, 85

Enablers
abstract, 243
for architecture, 245, 246
in the Big Picture, 243–247
capabilities, 244, 513
creating and managing, 244
cross-ART, 246–247
cross-value stream, 246–247
definition, 513
details, 243–247
epics, 244, 513
for exploration, 245
extending the architectural runway, 266–267
features, 244, 513
for infrastructure, 245–246
program level, 243–247
of solutions, 402
stories, 135–136, 244, 513. See also Stories.
types of, 244
uses for, 245–246

Enterprise
abstract, 417–418
in the Big Picture, 417–422
budget, 418, 421
business drivers, 421
competitive environment, 421
core values, 421
connecting to a portfolio, 418
definition, 513
details, 417–422
distinctive competence, 421–422
financial goals, 421
KPIs (key performance indicators), 420
mission, 421
multiple SAFe instances, 419–420
portfolio context, 418, 420–422
portfolio level, 417–422
portfolio strategy, 420–421
qualitative data, 420
sample portfolio, 418–419
strategic themes, 418
as a system, 52
vision, 421

Enterprise Architect, portfolio level, 439–442

Enterprise Architects
abstract, 439
architectural strategy, 440–442
in the Big Picture, 439–442
choice of technology, 441
definition, 513
details, 439–442

details, 243–247
development infrastructure strategy, 441

implementation strategy, 442

interprogram collaboration, 441–442

portfolio level, 439–442

roles and responsibilities, 439–440

working with System/Solution Architects, 441

Enterprise architecture, value stream coordination, 345–346

Enterprise Balanced Scorecard, 311–312

Enterprise SAFe, 41–42

Environment of mutual interest, motivating knowledge workers (SAFe Principle #8), 68–69

Epic Burn-up Chart, 309

Epic Owners

abstract, 435

in the Big Picture, 435–437

collaborative nature of, 437

definition, 514

details, 435–437

portfolio level, 435–437

roles and responsibilities, 435–437

Epic Progress Measure, 310

Epic specification workshop, 191

Epic Success Criteria, 310–311

Epic Value Statement Template, 484

Epics

abstract, 483

analysis, 484–485

approving investment in, 488

in the Big Picture, 483–488

definition, 514

details, 483–488

enablers, 244, 513

Epic Value Statement Template, 484

funding and governance, economic framework, 348

measuring progress, 486

metrics for, 309–311

portfolio Kanban, 444–445

portfolio level, 483–488

PPM (Program Portfolio Management), 432

program, 488, 516

splitting, 486–487

success criteria, 486

value stream, 488, 521

Estimating

Agile method, 433

features, 240

leveraging the estimation buffer, 251–252

portfolio backlog, 451–452

PPM (Program Portfolio Management), 433

relative, 116–117

Estimating poker, 138–140

Estimating workload

Agile Teams, 80

common starting point for estimation, 104

establishing velocity, 114

Kanban teams, 103–104

relative estimating, 116–117

stories. See Stories, estimating.

Evolving solution context, 406

Executable MBSE models, 363

Executives, training, 39

Expedite service class, 102–103

Exploration enablers, 245

Extending architectural runways, 266–267, 270

F

FAB (features and benefits) matrix, 238–239

Fan-out model for PO, PM, and Agile Teams, 86

FASB (Financial Accounting Standards Board), 465

FASB 86 guidelines, 465

Feature enablers, 244

Feature level CI (continuous integration), 491–492

Feature Progress Report, 314–315

Feature section, program Kanban, 190

Feature selection, program Kanban, 190

Features

abstract, 237

accepting, 240

in the Big Picture, 237–242

creating and managing, 239

definition, 237, 514

details, 237–242

splitting, 240

estimating, 240

FAB (features and benefits) matrix, 238–239

versus objectives, PI objectives, 227–228

prioritizing, 239

program level, 237–242

splitting capabilities into, 241–242

Feedback, balancing with integration effort, 235

Feedback, eliciting, 130

Financial Accounting Standards Board (FASB), 465

Financial goals, enterprise, 421

Firm fixed-price contracts, 503–505

Firmware, built-in quality, 151–152

The First Decision Rule Principle, 49
Fishbone diagrams, 256
Fitness for purpose, providing objective evidence for, 460. See also Objective milestones (SAFe Principle #5).
Five Whys technique, 256
Fixed date service class, 102–103
Fixed solution context, 406
Fixed solution intent, 354, 355–356
Fixed-date milestones, 327–328
Fixed-schedule programs, Set-Based Design, 366
Flow
architectural, implementing, 377
CFD (cumulative flow diagram), 101–102, 318
continuous value, PPM (Program Portfolio Management), 432
improving, iteration execution, 121–122
improving with Kanban, 102–103
measuring with Kanban, 101–102
primary keys to, 61. See also specific keys. *Principles of Product Development Flow*, 259–261
visualizing with Kanban, 100–101
workflow management, program level, 156
Forecasting
portfolio backlog, 451–452
with roadmaps, 303
Foundation layer, 2–3. See also Big Picture, SAFe foundation.
4-level SAFe, Big Picture, 1
Frequency of releases, 331–332
Frequent integration points, Set-Based Design, 367
Functional tests, test-first methods, 498, 501
Funnel stage, portfolio Kanban, 444, 445

G
Gemba (visiting the workplace), 29, 392, 442
Genchi genbutsu, 392
Go and see. See Gemba (visiting the workplace).
Goals, for sprints. See Iteration goals.
Governance, PPM (Program Portfolio Management), 433–434
Greenleaf, Robert, 169

H
Hardware, built-in quality, 24, 151–152
Honda, collaborating with suppliers, 392
Horizontal line-of-business organization, 17
House of Lean, 27, 28–30
Human factors. See UX (User Experience) design.

I
I&A (inspect and adapt) workshop. See also Relentless improvement.
abstract, 253
agreeing on problems, 255–256
in the Big Picture, 253–258
brainstorming, 257
Business Owners, 223–224
creating improvement backlog items, 257–258
definition, 514
details, 253–258
PI (Program Increment), 211
PI system demo, 254
problem-solving workshop, 255
program level, 253–258
Program Predictability Measure, 254
quantitative measurement, 254
restating problems, 257
retrospective, 255
root cause analysis, 256–257
at the value stream level, 258
workshop, 85
Impact analysis, MBSE (Model-Based Systems Engineering), 360–361
Implementation stage, portfolio Kanban, 445, 446–447
Implementation strategy, Enterprise Architects, 442
Implementing 1-2-3. See also Training.
abstract, 37–38
in the Big Picture, 37–42
continuous improvement, 41
definition, 514
details, 37–42
Enterprise SAFe, 41–42
pattern description, 3
training, 3
Implementing architectural flow, principle of Agile architecture, 377
Implementing SAFe 4.0 with SPC Certification, 38
In the Big Picture, web address, 271
Incremental build (SAFe Principle #4)
in the Big Picture, 57–58
cadence-based integration points, 58
integration points, creating, 57–58
overview, 57–58
Independent, Negotiable, Valuable, Estimable, Small, Testable (INVEST), 137
Independent NFRs, 203
Infrastructure, CI (continuous integration), 495
Infrastructure enablers, 245–246

Innovation and planning iteration
abstract, 249
advance development infrastructure, 251
in the Big Picture, 249–252
definition, 514
details, 249–252
enabling continuous learning, 251
integrating a complete solution, 250–251
leveraging the estimation buffer, 251–252
program level, 249–252
sample IP iteration calendar, 251–252
time for innovation, 250
time for PI events, 250

Inspect and adapt (I&A) workshop. See I&A (inspect and adapt) workshop.

Integrated learning cycles (SAFe Principle #4)
in the Big Picture, 57–58
faster learning through faster cycles, 58
integrated learning cycles, 60
PDCA (Plan-Do-Check-Adjust) cycle, 58
slipping integration points, 58

Integration, solutions, 403
Integration frequency, CI (continuous integration), 494

Integration points
cadence-based, 58
creating, 57–58
indication of a troubled project, 58
slipping, 58

Integration with other teams, Agile Teams, 80
Intense collaboration, Agile Teams, 78–79

Intentional architecture, 266, 371, 373
Interactive view of SAFe. See Big Picture.

Interfaces, specifying, 367

Internet of Things, DevOps, 274

Interprogram collaboration, Enterprise Architects, 441–442

INVEST (Independent, Negotiable, Valuable, Estimable, Small, Testable), 137

Ishikawa diagrams, 256

IT deployment environments, solution context, 407–408

Iteration execution. See also Program execution.
abstract, 119
in the Big Picture, 119–124
building stories, serially and incrementally, 122–124
built-in quality, 122
constant communication, 121
continuous story acceptance, 122
definition, 514
details, 119–124
DSU (daily stand-up) meeting, 121
improving flow, 121–122
intra-iteration waterfalls, 122–123
key elements of, 120
managing WIP, 121–122
PO (Product Owner), 84–85
program execution, 124
purpose of, 119–120
team level, 119–124
test automation, 122
tracking progress, 120–121
visualizing progress, 120–121

Iteration goals
abstract, 143
aligning program teams to common PI objectives, 145
aligning team members to a common purpose, 145
in the Big Picture, 143–145
continuous management information, 146
definition, 514
details, 143–146
importance of, 144–145
managing dependencies, 145
outputs of, 143–144
team level, 143–145

Iteration Metrics, 319

Iteration planning
abstract, 113
attendees, 115
in the Big Picture, 113–117
commitment, 115
definition, 514–515
details, 113–117
importance of, 114–115
managing dependencies, 114
guidelines, 116
innovation and planning iteration, 514
inputs to, 114
iteration goals, 115
normalizing story point estimating, 116–117
PO (Product Owner), 84
purpose of, 113–114
relative estimating, 116–117
sample agenda, 116
story analysis and estimation, 115
task analysis, 115
team level, 113–117

Iteration retrospective
abstract, 129
agenda, 131
in the Big Picture, 129–131
definition, 515
details, 129–131
eliciting feedback, 130
guidelines, 131
PO (Product Owner), 85
qualitative review, 130
quantitative review, 129–130
team level, 129–131
Iterations. See also PDCA cycles.
abstract, 109
adjusting, 111
in the Big Picture, 109–111
checking, 111
definition, 515
details, 109–111
executing, 110
PDCA cycle, 109–111
planning, 110
team level, 109–111

J
Jeffries, Ron, 136
Job duration, WSJF (Weighted Shortest Job First), 185–186
Job sequencing based on cost of delay, economic framework, 348, 349
Job size as a proxy for duration, WSJF (Weighted Shortest Job First), 186
Just-in-time story elaboration, PO (Product Owner), 84

K
Kanban. See also Team Kanban.
average lead time, 101–102
CFDs (Cumulative Flow Diagrams), 101–102
classes of service, 102–103
description, 100
improving flow, 102–103
measuring flow, 101–102
Portfolio Kanban Board, 308–309
Program Kanban Board, 315
as a pull system, 99
swim lanes, 102–103
Team Kanban Board, 320
throughput, 101–102
Value Stream Kanban Board, 313
visualizing flow, 100–101
WIP limits, 100–101
Kettering, Charles, 255
KPIs (key performance indicators)
enterprise, 420
PPM (Program Portfolio Management), 433

L
Labor efforts, as CapEx (capital expense), 469–471
Lazier, William, 420
Leaders, training, 39

Leading SAFe 4.0, Leading the Lean-Agile Enterprise with the Scaled Agile Framework, 39
Lean Portfolio Metrics, 307–308
The Lean Startup, 420
Lean-Agile approach to System Architect/Engineering, 173–175
Lean-Agile budgeting, economic framework, 348
Lean-Agile Budgeting: Beyond Project Cost Accounting, 454–455
Lean-Agile change agents, training, 38, 39
Lean-Agile Leaders. See also specific leaders.
abstract, 11
in the Big Picture, 11–15
definition, 2, 515
details, 11–15
philosophy of SAFe, 11
in System Architect/Engineering, 174
traits of, 174
Lean-Agile Leaders, desirable behaviors
decentralized decision-making, 12
developing people, 12
emphasizing lifelong learning, 12
leading change, 11
minimizing constraints, 12
motivating knowledge workers, 12
Lean-Agile mindset. See also Agile Manifesto.
abstract, 27
Big Picture, 27–32
definition, 515
details, 27–33
foundation of, 30
leadership philosophy, 30
training leaders in, 3
Lean-Agile mindset, House of Lean
definition, 27
flow, 29
four pillars, 28–30
goal, 28
innovation, 29–30
relentless improvement, 30, 81
respect for people and culture, 28–29
Lean-Agile practices, suppliers, 390
Learning cycles, MBSE models, 360
Learning milestones, 326–327
Licensing, 38
Life-cycle governance, PPM (Program Portfolio Management), 433–434
Lightweight business cases, PPM (Program Portfolio Management), 432
Little's Law
- program backlogs, 197
- queue length, 62
- value stream backlogs, 197

Management, role in changing systems, 52–53
Managers, training, 39
Managing queue lengths (SAFe Principle #6)
- in the Big Picture, 61–62
- flow limit, 62
- Little's Law, 62
- queue length versus wait time, 62
Man-machine interface. See UX (User Experience) design.

Marick, Brian, 497
MBSE (Model-Based Systems Engineering)
- abstract, 359
- in the Big Picture, 359–364
- built-in quality, 362
- compliance, 360–361, 362
- definition, 515
- details, 359–364
- documentation, 362, 364
- executable models, 363
- impact analysis, 360–361
- model standards, 362–363
- models and learning cycles, 360
- recording models in solution intent, 361–362
- testable models, 363
- tool for solution intent, 203
- traceability, 360–361
- value stream level, 359–364

Measurements. See Metrics; Milestones.
Metrics. See also Milestones.
- abstract, 307
- ART Self-Assessment, 318–319
- in the Big Picture, 307–321
- CFD (cumulative flow diagram), 318
- definition, 515
- details, 307–321
- Enterprise Balanced Scorecard, 311–312
- Epic Burn-up Chart, 309
- Epic Progress Measure, 310
- Epic Success Criteria, 310–311
- for epics, 309–311
- Feature Progress Report, 314–315
- Iteration Metrics, 319
- Lean Portfolio Metrics, 307–308
- measuring epic progress, 486
- measuring progress against strategic themes, 426–427
- objective milestones (SAFe Principle #5), 60
- PI Burn-down Chart, 317–318
- Portfolio Kanban Board, 308–309
- Portfolio Metrics, 307–308
- for portfolios, 307–312
- Program Kanban Board, 315
- Program Performance Metrics, 316–317
- Program Predictability Measure, 316
- for programs, 314–319
- SAFe Team Self-Assessment, 321
- spanning palette, 307–321
- Team Kanban Board, 320
- Team PI Performance Report, 320–321
- for teams, 319–321
- Value Stream Kanban Board, 313
- Value Stream Performance Metrics, 314
- Value Stream Predictability Measure, 313–314
- for value streams, 313–314

Milestones. See also Metrics.
- abstract, 323
- in the Big Picture, 323–329
- definition, 515
- details, 323–329
- fixed-date, 327–328
- learning, 326–327
- measuring success, 328–329
- object measurement. See Objective milestones (SAFe Principle #5).
- phase-gate (waterfall), 59–60, 324
- PI (Program Increment), 325–326
- planning and executing, 328
- spanning palette, 323–329

Minimal constraints, motivating knowledge workers (SAFe Principle #8), 68

Mission
- enterprise, 421
- motivating knowledge workers (SAFe Principle #8), 68

“Mixing Agile and Waterfall Development,” 158, 391

Mock, 495

Model-Based Systems Engineering (MBSE). See MBSE (Model-Based Systems Engineering).

Modeling, Set-Based Design, 367

Modeling alternative approaches, principle of Agile architecture, 375–376

Models, MBSE and learning cycles, 360
- recording in solution intent, 361–362
- standards, 362–363
- testable, 363
- traceability, 360–361

Moore, Geoffrey, 420

Motivating knowledge workers (SAFe Principle #8)
- autonomy, 68
- in the Big Picture, 67–69
decision-making framework. See Decentralizing decision-making (SAFe Principle #9).

drive, 68
Drucker on, 67
environment of mutual interest, 68–69
leveraging the systems view, 67
minimal constraints, 68
mission, 68
purpose, 68
the role of compensation, 67

Multi-ART value streams, 211, 219–220, 480
Multiple value stream ARTs, 478–479
Murphy’s Law, DevOps, 275

Negotiable NFRs, 203
NFRs (nonfunctional requirements)
abstract, 199
all-at-once approach, 204
as backlog constraints, 201–202
in the Big Picture, 199–206
bounded, 203
criteria for, 203
definition, 515
details, 199–206
economic impacts of, 202
fitness for use, 199–200
FURPS (Functionality, Usability, Reliability, Performance, Supportability), 199–200
implementation approaches, 203–204
independent, 203
negotiable, 203
program level, 199–206
at SAFe levels, 200–201
solution intent, 202–203
and solutions, 402
specifying, 203
story-by-story approach, 204
system qualities tests, 499
systemic impacts of, 202
testable, 203
testing, 204–206, 499
tools for solution intent, 203

Objective milestones (SAFe Principle #5)
in the Big Picture, 59–60
cadence (SAFe Principle #7), 60
incremental build, integrated learning cycles (SAFe Principle #4), 60
measurement frequency, 60
metrics, 60
versus phase-gate milestones, 59–60, 324

Objectives, business. See Strategic themes.

Objectives versus features, PI objectives, 227–228
Ohno, Taiichi, 29
Online resources. See Web addresses; Websites.
Oosterwal, Dantar P., 324
Operating CoPs (Communities of Practice), 18–19
Operational value streams, 474–475
OpEx (operational expense), definition, 512
OpEx (operational expense), portfolio level, 463–471
abstract, 463–464
Agile development capitalization strategies, 467–468
allowable labor efforts, 470–471
applying stories, 468–470
in the Big Picture, 463–471
capitalization triggers in waterfall development, 466–467
capitalization versus expense criteria, 465–466
capturing labor effort, 469
categorization of features, 468
details, 463–471
FASB (Financial Accounting Standards Board), 465
FASB 86 guidelines, 465
portfolio level, 463–471
software classification, 465
software development costs, 465
by story count, 470
by story hours, 469
by story points, 469–470
Optimizing life cycle profits, economic view (SAFe Principle #1), 48–49
Organizing CoPs (Communities of Practice), 18
Overview of SAFe, interactive. See Big Picture.

Pair programming, built-in quality, 150
Pair work, built-in quality, 150
Pareto Analysis, 256–257
PDCA (Plan-Do-Check-Adjust) cycles. See also PI (Program Increment).
adjusting the next iteration, 111
checking iterations, 111
executing iterations, 110
factor in faster learning, 58
in iterations, 109–111
planning iterations, 110. See also Innovation and planning iteration.
Personnel and team development, development manager, 13–14
Phase-gate (waterfall) milestones, 59–60, 324
PI (Program Increment). See also PDCA (Plan-Do-Check-Adjust) cycle.
abstract, 207
in the Big Picture, 207–212
definition, 517
develop on cadence, release any time, 208
executing, 208–212
I&A (inspect and adapt) workshop, 211
milestones, 325–326
multi-ART value streams, 211
PI planning, 209, 211
PO Sync meeting, 210
pre- and post-PI planning, 211
program level, 207–212
solution demo, 211–212
SoS (Scrum of Scrums) meetings, 210
system demo, 211
value stream increment, 211–212
PI Burn-down Chart, 317–318
PI cadence, portfolio Kanban, 447
PI execution, responsibilities of Business Owners, 223–224
PI objectives
aligning program teams, iteration goals to, 145
program level, 225–231
team level, definition, 520
value stream level, definition, 521
PI objectives, program level
abstract, 225
assigning business values to objectives, 229–230
in the Big Picture, 225–231
committing to, 230
definition, 517
details, 225–231
features versus objectives, 227–228
program objectives, creating, 230–231
shedding excess WIP, 231
SMART objectives, 229
stretch objectives, 228–229
team PI objectives, finalizing, 230
team PI objectives, setting, 227
value stream objectives, creating, 230–231
PI planning event. See also Pre- and post-PI planning.
abstract, 213
agenda, 216–218
Agile Teams, 80
in the Big Picture, 213–220
business benefits, 214
definition, 215
details, 213–220
facility readiness, 215
inputs to, 214
in multi-ART value streams, 219–220
outputs, 218–219
PI (Program Increment), 209, 211
PO (Product Owner), 84
post-event activities, 219
preparation for, 214–218
program backlogs, 195
program level, 213–220
Release (PI) Planning event, 64–65
Release Planning Readiness Checklist, 215
responsibilities of Business Owners, 222
sample program board, 218–219
value stream backlogs, 195
value stream Kanban, 191
PI planning report, 386–387
PI summary reports, 385
PI system demo, I&A (inspect and adapt) workshop, 254
Pillars of the House of Lean, 28–30. See also specific pillars.
Pink, Daniel, 67–68
Plan review, pre- and post-PI planning meetings, 387
Plan-Do-Check-Adjust (PDCA) cycles. See PDCA (Plan-Do-Check-Adjust) cycles.
Planning. See also Iteration planning; PI planning event;
Pre- and post-PI planning meetings.
innovation and planning iteration, 514
milestones, 328
PPM (Program Portfolio Management), 433
PM (Product Management)
abstract, 177
ART (Agile Release Train), 161
in the Big Picture, 177–181
definition, 516
details, 177–181
program level, 177–181
responsibilities, 179–181
PO (Product Owner). See also ScrumXP teams; SM (Scrum Master).
abstract, 83
accepting stories, 84
ATDD support, 84
backlog refinement, 84
in the Big Picture, 83–87
content authority, 85–86
definition, 516
details, 83–86
fan-out model for Product Management and Agile Teams, 86
inspect and adapt workshop, 85
iteration execution, 84–85
iteration planning, 84
iteration retrospective, 85
just-in-time story elaboration, 84
in PI planning, 84
program execution, 85
responsibilities, 83–86
scope of enabler work, 85
team demos, 85
training, 41
PO Sync meeting, 210
Point-Based Design, versus Set-Based Design, 365–366
Portfolio business epics, definition, 516
Portfolio context
enterprise, 418, 420–422
PPM (Program Portfolio Management), 430–431
Portfolio Kanban
abstract, 443
analysis stage, 445, 446
in the Big Picture, 443–447
definition, 516
details, 443–447
done stage, 445, 447
for epics, 444–445
funnel stage, 444, 445
implementation stage, 445, 446–447
PI cadence, 447
portfolio backlog stage, 445, 446
portfolio level, 443–447
purpose of, 444
review stage, 445, 446
Portfolio Kanban Board, 308–309
Portfolio level
abstract, 413
in the Big Picture. See Big Picture, portfolio level.
budgets, 453–461
CapEx (capital expense). See CapEx (capital expense), portfolio level.
connection to the enterprise, 414
definition, 2, 516
description, 6–7
details, 413–415, 423–427
enterprise, 417–422
Enterprise Architect, 439–442
Enterprise Architect, duties of, 415
Epic Owners, 435–437
epics, 483–488
introduction, 413–415
key concepts, 414–415
managing epic workflow, 414–415
OpEx (operational expense). See OpEx (operational expense), portfolio.
portfolio backlog, 449–452
portfolio Kanban, 443–447
PPM (Program Portfolio Management), 414, 429–434
strategic themes, 423–427
value streams, 473–482
web address, 411
Portfolio management. See PPM (Program Portfolio Management).
Pre- and post-PI planning meetings
abstract, 383
in the Big Picture, 383–388
business benefits, 384
business context, 385
certainty vote, 387
definition, 516
details, 383–388
inputs to, 384
next PI features, 386
output, 387–388
PI planning report, 386–387
PI summary reports, 385
plan review, 387
preparing for, 211, 385
retrospective planning, 387
rework, planning, 387
risk analysis, 387
setting planning context, 385–386
solution context, 385
solution demos, 384
stakeholder participation, 386
summarizing results, 386–387
value stream backlog, 386
value stream level, 383–388
Preserving options (SAFe Principle #3)
in the Big Picture, 55–56
creating integration points, 57–58
overview, 55–56
The Principle of Continuous Economic Trade-Offs, 49
The Principle of Optimum Decision Making, 49
The Principle of Quantified Cost of Delay, 49
Principles
Agile architecture. See Agile architecture, basic principles.
Agile Manifesto, 32
Lean. See SAFe principles.
for making economic decisions, 49, 195
SAFe. See SAFe principles.
Principles of Product Development Flow, 259–261
Priorities, strategic themes, 426
Prioritizing
features, 239
jobs, 183–186
program backlogs, 194
value stream backlogs, 194
Problem-solving workshop, 255. See also I&A (inspect and adapt) workshop.
Product Management (PM). See PM (Product Management).
Product Owner (PO). See PO (Product Owner).
Product Owners, team level, 83–87
Program backlogs
abstract, 193
in the Big Picture, 193–197
capacity allocation, 195–196
definition, 516
details, 193–197
Little's Law, 197
PI planning, 195
prioritizing, 194
program level, 193–197
queues, 197
refining, 194
solution integrity, optimizing, 195–196
value, optimizing, 195–196
wait times, 197
Program board, sample, 218–219
Program epic section, program Kanban, 189
Program epics
definition, 516
exploring and approving with program Kanban, 189
Program execution. See also Iteration execution.
core value of SAFe, 25
development manager, 14
iteration execution, 124
PO (Product Owner), 85
at the program level, 156
Program Increment (PI). See PI (Program Increment).
Program Kanban
abstract, 187
in the Big Picture, 187–191
definition, 517
details, 187–191
exploring and approving program epics, 189
feature section, 190
feature selection, 190
program epic section, 189
program level, 187–191
Program Kanban Board, 315
Program level
abstract, 155
architectural runway, 265–270
ART (Agile Release Train), 159–165
in the Big Picture. See Big Picture, program level.
Business Owners, 221–224
capabilities, 237–242
definition, 2, 517
description, 4–5, 155–158
details, 155–158
develop on cadence, release any time, 259–264
enablers, 243–247
features, 237–242
I&A (inspect and adapt), 253–258
innovation and planning iteration, 249–252
introduction, 155–158
key roles, 156–157
managing workflow, 156
NFRs (nonfunctional requirements), 199–206
PI (Program Increment), 207–212
PI objectives, 225–231
PI planning, 213–220
portfolio, connection to, 157
Product Management, 177–181
program backlogs, 193–197
program Kanban, 187–191
RTE (Release Train Engineer), 167–170
Solution Architect/Engineering, 171–175
Solution Management, 177–181
System Architect/Engineering, 171–175
system demo, 233–236
value stream, connection to, 157
value stream backlogs, 193–197
value stream Kanban, 187–191
VSE (Value Stream Engineer), 167–170
web address, 153
WSJF (Weighted Shortest Job First), 183–186
Program level epics, 488
Program management, 345–346. See also PPM (Program Portfolio Management).
Program Performance Metrics, 316–317
Program PI objectives, definition, 517
Program Portfolio Management (PPM). See PPM (Program Portfolio Management).
Program Predictability Measure, 254, 316
Program teams, aligning to common PI objectives, 145
Program Vision, 299
Programs, metrics for, 314–319
Program/value stream backlog refinement, value stream Kanban, 191
Progress tracking. See Tracking progress.
visualizing. See Visualizing progress.
Project-based constraints, effects on, 456–457
Promise for a conversation, 136
Prototyping, Set-Based Design, 367
Pugh, Ken, 500
Pull systems, Kanban, 99
Purpose, motivating knowledge workers (SAFe Principle #8), 68
Qualitative review, iteration retrospective, 130
Quantitative measurement, I&A (inspect and adapt) workshop, 254
Quantitative review, iteration retrospective, 129–130
Queues
avoiding excessive length, 305–306
managing. See Managing queue lengths (SAFe Principle #6).
program backlogs, 197
value stream backlogs, 197
versus wait time, 62
The real place. See Gemba (visiting the workplace).
Recording models in solution intent, Agile architecture, 376
Reduce batch sizes (SAFe Principle #6), 61–62
Refactoring, built-in quality, 150
Refining backlogs, 107–108, 194
Regression tests, 494
Reinertsen, Donald
on decision-making, 49
Principles of Product Development Flow, 259–261
sunk cost principle, 195
Release any time. See also Develop on cadence, release any time.
in the Big Picture, 259–264
definition, 517
Release Management
abstract, 283
in the Big Picture, 283–285
components of, 283
definition, 517
details, 283–285
membership of, 283–284
responsibilities for, 283–284
spanning palette, 283–285
weekly meetings, 285
Release (PI) Planning event, 64–65. See also PI planning.
Release Planning Readiness Checklist, 215
Release Train. See ARTs (Agile Release Trains).
Release Train Engineer (RTE). See RTE (Release Train Engineer).
Releases
abstract, 331
in the Big Picture, 331–336
definition, 517
definition of done, 335–336
details, 331–336
develop on cadence, release any time, 332
frequency, 331–332
spanning palette, 331–336
System Team roles and responsibilities, 281
value stream coordination, 346
Releases, building
actual release, 334–335
preventing release issues, 335
solution increment, 334
system increment, 333–334
team increment, 333
Releasing any time, developing on cadence, 263–264
Relentless improvement. See also I&A (inspect and adapt) workshop.
Agile Teams, 81
backlog items, creating, 257–258
Implementing 1-2-3, 41
Lean-Agile mindset, House of Lean, 30, 81
Requirements, specifying, 367
Responding to change over following a plan, roadmaps, 301–302
Responsibilities. See Roles and responsibilities.
Retrospective
I&A (inspect and adapt) workshop, 255
planning, pre- and post-PI planning meetings, 387
Review stage, portfolio Kanban, 445, 446
Rework, planning, 387
Risk analysis, pre- and post-PI planning meetings, 387
Risk reduction-opportunity enablement value, 184
Roadmaps
abstract, 301
avoiding long queues, 305–306
in the Big Picture, 301–306
building, 302–303
definition, 518
details, 301–306
estimating longer-term initiatives, 304–305
example, 302
Little's Law, 306
long-term forecasting, 303
responding to change over following a plan, 301–302
spanning palette, 301–306
view of Vision, 299–300
Roles and responsibilities
Agile Teams, 78
Business Owners, 221–224
development manager, 13
Enterprise Architects, 439–440
Epic Owners, 435–437
PM (Product Management), 179–181
RTE (Release Train Engineer), 167–168
Shared Services, 288–289
SM (Scrum Master), 89–91
Solution Architect/Engineering, 172–173
Solution Management, 181
System Architect/Engineering, 172–173
UX (User Experience), 291–292
VSE (Value Stream Engineer), 167–168
Roles and responsibilities, System Team
details, 279–282
release, 281
solution demos, 281
solution performance testing, 281
system demos, 281
system integration, 280
Root cause analysis
80/20 rule, 256–257
fishbone diagrams, 256
Five Whys technique, 256
I&A (inspect and adapt) workshop, 256–257
Ishikawa diagrams, 256
Pareto Analysis, 256–257
RTE (Release Train Engineer)
abstract, 167
ART (Agile Release Train), 161
in the Big Picture, 167–170
definition, 517
details, 167–170
program level, 167–170
reporting structure, 169
responsibilities, 167–168
as servant leaders, 169
value stream Kanban, 191
Runway. See Architectural runway.

5
SAFe (Scaled Agile Framework)
interactive view of. See Big Picture.
organization levels, 2. See also Spanning palette; specific levels.
websites. See Websites, SAFe (Scaled Agile Framework).
SAFe 4.0 Advanced Scrum Master with ASM certification, 41
SAFe 4.0 Product Manager / Product Owner with PMPO certification, 41
SAFe 4.0 Scrum Master Orientation, 41
SAFe 4.0 with SAFe Program Consultant (SPC4) Certification, 39
SAFe foundation, 2–3. See also Big Picture, SAFe foundation.
SAFe Managed-Investment Contracts, 506–509
SAFe principles
in the Big Picture. See Big Picture, SAFe principles.
definition, 518
importance of, 36
list of, 3, 35, 43. See also specific principles.
SAFe Team Self-Assessment, 321
Scaled Agile Framework (SAFe). See SAFe (Scaled Agile Framework).
Scaled Agile SPC membership site, 39
Scaling
alignment, 22
crappy code/components, 23–24
Scrum Master (SM). See SM (Scrum Master).
Scrum of Scrums (SoS), 210
Scrum process
BVIRs (big visual information radiators), 95
dSU (daily stand-up) meeting, 95
iteration planning, 94–95
iteration retrospective, 96
iterations versus sprints, 94
stories, dividing into tasks, 94–95
story boards, 95
team demo, 96
tracking progress, 95
visualizing work, 95
ScrumXP teams
abstract, 93
on the ART train, 96–97
in the Big Picture, 93–97
definition, 518
details, 93–97
leadership, 97
PO (Product Owner), 94
SM (Scrum Master), 94
team level, 93–97
Set-Based Design
abstract, 365
adaptive planning, 368
in the Big Picture, 365–369
definition, 518
details, 365–369
economic efficiency, 366–368
economic trade-offs, 368–369
fixed-schedule programs, 366
frequent integration points, 367
modeling, 367
versus Point-Based Design, 365–366
prototyping, 367
recommended practices, 367–368
simulation, 367
specifying interfaces and requirements, 367
tool for solution intent, 203
value stream level, 365–369
Shared Services
abstract, 287
ART (Agile Release Train), 161
in the Big Picture, 287–289
definition, 518
details, 287–289
embedded in Agile teams, 289
roles and responsibilities, 288–289
skills required, 288
spanning palette, 287–289
specialized training, 288
Simplicity, principle of Agile architecture, 375
Simulation, Set-Based Design, 367
Single value stream ARTs, 479–480
Skill requirements, Agile Teams, 77–78
Slipping integration points, 58
SM (Scrum Master)
abstract, 89
in the Big Picture, 89–91
CoPs (Communities of Practice), 18–19
definition, 518
details, 89–91
responsibilities, 89–91
sourcing the role, 91
team level, 89–91
time requirements, 91
training, 41
SMART objectives, 229
Software, built-in quality, 23, 148–151
Software classification, CapEx (capital expense), 465
Software development costs, CapEx (capital expense), 465
Solution Architect/Engineering
abstract, 171
in the Big Picture, 171–175
definition, 518
details, 171–175
origin of SAFe roles, 173
program level, 171–175
responsibilities, 172–173
Solution context
abstract, 405
in the Big Picture, 405–409
continuous collaboration, 408–409
definition, 518
deployability, 408–409
details, 405–409
driving solution intent, 406
fixed versus evolving, 406
for IT deployment environments, 407–408
portfolio-level concerns, 408
pre- and post-PI planning meetings, 385
synchronization with CI (continuous integration), 493–494
for a system of systems, 407
types of, 407
value stream level, 405–409

Solution demos
abstract, 379
agenda, 381
attendees, 380
in the Big Picture, 125, 379–382
definition, 519
demonstrating the solution, 381
details, 379–382
guidelines, 381
investment, 382
PI (Program Increment), 211–212
pre- and post-PI planning meetings, 384
preparation for, 380
as a pull event, 380
strategy, 382
System Team roles and responsibilities, 281
team level, 125
timing, 382
value stream level, 379–382

Solution increment, 334

Solution integration
CI (continuous integration), 492–494
versus testing effort, 281–282

Solution integrity, optimizing, 195–196

Solution intent
abstract, 351
Agile Development in High-Assurance and Regulated Environments, 352
assuming variability, 353
in the Big Picture, 351–358
capturing portfolio Vision, 298–299
collaborating on, 354–355
current and future state, 352–353
definition, 519
designs, 352–353
details, 351–358
developing, 354–355
documentation, 357–358
driven by solution context, 406
dynamic nature of, 353
fixed, 354, 355–356
introduction, 352–353
and NFRs (nonfunctional requirements), 202–203
and NFRs, 402
and recording MBSE models, 361–362
recording models in, Agile architecture, 376
and solutions, 403
specifications, 352–353
system level, 356–357
tests, 352–353
value stream level, 351–358
variable, 354, 355–356

Solution Management
abstract, 177
in the Big Picture, 177–181
definition, 519
details, 177–181
program level, 177–181
responsibilities, 181

Solution performance testing, System Team roles and responsibilities, 281

Solution vision, 297–298

Solutions
abstract, 401
in the Big Picture, 401–404
capabilities, 402
customer context, 403
definition, 518
demos, 403
details, 401–404
development overview, 401–402
economic viability, 403
enablers, 402
integration, 403
multiple, managing, 403–404
NFRs, 402
solution intent, 403
as systems, 51–52
systems thinking, 402–403
testing, 403
value stream level, 401–404

Solution/system demo, value stream Kanban, 191

SoS (Scrum of Scrums), 210

Spanning palette
in the Big Picture. See Big Picture, spanning palette.
definition, 519
description, 5
DevOps, 273–278
metrics, 307–321
milestones, 323–329
at the program level, 157
Release Management, 283–285
releases, 331–336
roadmaps, 301–306
Shared Services, 287–289
System Team, 279–282
UX (User Experience), 291–294
Vision, 295–300
web address, 271

Specialist roles, training, 41

Specification workshops, team backlog, 108
Specifications, solution intent, 352–353

Splitting
capabilities into features, 241–242
epics, 486–487
stories, 140
value streams, 163–165

Sprint goals. See Iteration goals.

Sprints. See Iterations.

Stakeholder participation, pre- and post-PI planning meetings, 386

Standard service class, 102–103

Statement of intent, 136

Stories
3Cs: Card, Conversation, Confirmation, 136–137
abstract, 133
acceptance criteria, 137
accepting, 84
analysis of, 115
applying to CapEx, 468–470
in the Big Picture, 133–141
building serially and incrementally, 122–124
Card aspect, 136
Confirmation aspect, 137
continuous acceptance, 122
Conversation aspect, 136–137
definition, 519
details, 133–141
enabler, 135–136, 513
guidelines for writing, 136–137
hierarchy of artifacts, 133–134
INVEST (Independent, Negotiable, Valuable, Estimable, Small, Testable), 137
promise for a conversation, 136
in the SAFe Requirements Model, 141
sources of, 134
splitting, 140
statement of intent, 136
task analysis, 115
team level, 133–141
user, 134
value centric, 134

Stories, estimating
common starting baseline, 139–140
estimating poker, 138–140
iteration planning, 115
normalizing story point estimating, 116–117
story points, 138
velocity, 139

Story boards, 95

Story count, CapEx (capital expense), 470

Story enablers, 244

Story hours, CapEx (capital expense), 469

Story points
CapEx (capital expense), 469–470
estimating stories, 138

Story-by-story approach to NFRs, 204

Strategic themes
abstract, 423
in the Big Picture, 423–427
connecting enterprise to portfolio, 418
definition, 519
economic framework, 425
examples, 424
formulating, 424–425
input to portfolio vision, 425–426
measuring progress against, 426–427
portfolio backlog, 426
portfolio level, 423–427
priorities, 426
value streams, 425–426
vision, 426

Stretch objectives, 228–229

Stubs, 495

Subsystem ARTs, 163–164

Success criteria for epics, 486

The Sunk Cost Principle, 49

Suppliers
abstract, 389
Agile contracts, 393
in the Big Picture, 389–394
collaborating with, 392–393
decentralizing decision-making, 391–392
definition, 519
details, 389–394
improving, 393
Lean-Agile practices, 390
selecting, 393
synchronization with CI (continuous integration), 493–494
systems thinking, 391–392
traditional methods, 391
value stream level, 389–394

Swim lanes, 102–103

Synchronization
with cross-domain planning, 63–65
develop on cadence, release any time, 259–261
value stream coordination, 344–345

System architect, origin of SAFe roles, 173

System Architect/Engineering
abstract, 171
ART (Agile Release Train), 161
in the Big Picture, 171–175
decentralized decision-making, 174
definition, 519
details, 171–175
empirical approach, 174–175
Lean–Agile approach, 173–175
origin of SAFe roles, 173
program level, 171–175
at the program level, 157
responsibilities, 172–173
traits of Lean–Agile Leaders, 174

System demos. See also Team demos.
abstract, 233
agenda, 235
attendees, 236
balancing integration effort and feedback, 235
in the Big Picture, 125, 233–236
definition, 520
details, 233–236
PI (Program Increment), 211
program level, 233–236
System Team roles and responsibilities, 281
team level, 125
timing of, 234–235
System health, optimizing, 107
System increment of releases, 333–334
System integration
built-in quality, 24
System Team roles and responsibilities, 280
System Kanban, details, 187–191
System level, solution intent, 356–357
System of systems, solution context, 407
System qualities. See NFRs (nonfunctional requirements).
System qualities tests, 499

System Team
abstract, 279
ART (Agile Release Train), 161
in the Big Picture, 279–282
building development infrastructure, 280
CI (continuous integration) in support of, 496
definition, 520
details, 279–282
in larger value streams, 280
release, 281
roles and responsibilities, 279–282
solution integrations versus testing effort, 281–282
solution performance testing, 281
spanning palette, 279–282
system and solution demos, 281
system integration, 280
Systems engineer, origin of SAFe roles, 173

Systems thinking (SAFe Principle #2)
in the Big Picture, 51–53
the Enterprise is a system, 52
management role in changing systems, 52–53
the solution is a system, 51–52
solutions, 402–403
suppliers, 391–392

T
TDD (Test-Driven Development), 149. See also Test-first methods.

Team backlogs. See also Backlogs.
abstract, 105
in the Big Picture, 105–108
capacity allocation, 107
definition, 520
details, 105–108
refinement, 107–108
specification workshops, 108
system health, optimizing, 107
team level, 105–108
value delivery, optimizing, 107

Team demos
abstract, 125
agenda, 127
attendees, 127
in the Big Picture, 125–127
definition, 520
details, 125–127
functions of, 125
guidelines, 127
PO (Product Owner), 85
process, 126
purpose of, 126
team level, 125–127

Team Kanban. See also Kanban.
abstract, 99
on the ART train, 103
in the Big Picture, 99–104
calculating derived velocity, 104
common starting point for estimation, 104
definition, 520
details, 99–104
estimating work, 103–104
team level, 99–104

Team Kanban Board, 320

Team level
Agile Teams, 77–81
in the Big Picture. See Big Picture, team level.
built-in quality, 147–152
definition, 2, 520
demos, 125–127
description, 3–4
iteration execution, 119–124
iteration goals, 143–145
iteration planning, 113–117
iteration retrospective, 129–131
iterations, 109–111
Product Owners, 83–87
purpose of, 3
Scrum Master, 89–91
ScrumXP teams, 93–97
solution demo, 125
stories, 133–141
system demo, 125
team backlog, 105–108
team demo, 125–127
team Kanban, 99–104
Team members, DevOps, 274
Team PI objectives, definition, 520
Team PI Performance Report, 320–321
Teams. See also specific teams.
metrics for, 319–321
on the train, 164–165
“Technical Strategies for Agile and Waterfall Interoperability at Scale,” 391
Test automation, iteration execution, 122
Test environments, DevOps, 276
Testable MBSE models, 363
Testable NFRs, 203
Test-Driven Development (TDD), 149. See also Test-first methods.
Test-first methods. See also ATDD (Acceptance Test-Driven Development); TDD (test-driven development).
acceptance test template/checklist, 502
Agile Testing Matrix, 497–499
ATDD (Acceptance Test-Driven Development), 149,
automated acceptance testing, 502
in the Big Picture, 497–502
built-in quality, 149
component tests, 498, 500
definition, 520
details, 497–502
functional tests, 498, 501
recommended practices, 499–502
system qualities tests, 499
system-level acceptance tests, 498–499
TDD (Test-Driven Development), 149
unit tests, 498, 500
Testing
deprecated tests, 494
designing for testability, 496
NFRs (nonfunctional requirements), 204–206
partial, 205
regression tests, 494
responsibility for, 376–377
solutions, 403
UX design, 293
Testing effort versus solution integrations, 281–282
Tests, solution intent, 352–353
3Cs: Card, Conversation, Confirmation, 136–137
3-level SAFe, Big Picture, 1
Throughput, Kanban, 101–102
Time and materials contracts, 503–505
Time criticality, cost of delay factor, 184
Time to market, reducing, 481–482
Tools, for solution intent, 203
Toyota, 392
Traceability, MBSE models, 360–361
Tracking progress. See also Kanban.
BVIRs (big visual information radiators), 95
DSU (daily stand-up) meeting, 95, 121
iteration execution, 120–121
story boards, 95
Training
Agile implementers, 38
doing the ART (Agile Release Train), 40
consulting activities, 40–41
executives, 39
implementing 1-2-3 pattern, 3
Implementing SAFe 4.0 with SPC Certification, 38
leaders, 39
Leading SAFe 4.0, Leading the Lean-Agile Enterprise with the Scaled Agile Framework, 39
Lean-Agile change agents, 38, 39
licensing, 38
managers, 39
Product Owners, 41
SAFe 4.0 Advanced Scrum Master with ASM Certification, 41
SAFe 4.0 Product Manager / Product Owner with PMPO certification, 41
SAFe 4.0 Scrum Master Orientation, 41
SAFe 4.0 with SAFe Program Consultant (SPC4) Certification, 39
Scrum Masters, 41
specialist roles, 41
Transparency
core value of SAFe, 24–25
development manager, 14
enabling trust, 24–25

U
Uniform Resource Locators (URLs). See Web addresses; Websites.
Unit tests, test-first methods, 498, 500
URLs (Uniform Resource Locators). See Web addresses; Websites.
Usability. See UX (User Experience) design.
User Experience (UX) design. See UX (User Experience) design.
User stories, 134
User-business value, cost of delay factor, 184
UX (User Experience) design
abstract, 291
in the Big Picture, 291–294
centralized guidance and implementation, 293–294
characteristics of, 292
definition, 520
design criteria, 293
designers on the ART, 161, 293–294
details, 291–294
distributed, governed development, 294
interfaces, specifying, 367
as potential bottleneck, 294
roles and responsibilities, 291–292
spanning palette, 291–294
testing criteria, 293
Value, optimizing, 195–196
Value centric stories, 134
Value delivery, optimizing, 107
Value stream, importance of DevOps, 273–274
Value stream backlogs
abstract, 193
in the Big Picture, 193–197
capacity allocation, 195–196
definition, 516, 521
details, 193–197
Little's Law, 197
PI planning, 195
pre- and post-PI planning meetings, 386
prioritizing, 194
program level, 193–197
queues, 197
refining, 194
solution integrity, optimizing, 195–196
value, optimizing, 195–196
wait times, 197
Value stream coordination
abstract, 343
in the Big Picture, 343–346
cadence, 344–345
content management, 345–346
definition, 521
deployment, 346
details, 343–346
enterprise architecture, 345–346
new portfolio work levels, 345
portfolio roadmap, 346
program management, 345–346
release, 346
synchronization, 344–345
value stream level, 343–346
Value Stream Engineer (VSE). See VSE (Value Stream Engineer).
Value stream epics, definition, 521
Value stream increment, PI (Program Increment), 211–212
Value stream Kanban
in the Big Picture, 187–191
definition, 521
epic specification workshop, 191
PI planning, 191
program level, 187–191
program/value stream backlog refinement, 191
solution/system demo, 191
supporting ceremonies, 191
Value Stream Kanban Board, 313
Value stream level
Agile architecture, 371–378
in the Big Picture. See Big Picture, value stream level.
coordination of dependencies, 343–346
Customers, 395–399
definition, 2, 521
description, 6
economic framework, 347–349
MBSE (Model-Based Systems Engineering), 359–364
pre- and post-PI planning, 383–388
Set-Based Design, 365–369
solution context, 405–409
solution demo, 379–382
solution intent, 351–358
solutions, 401–404
suppliers, 389–394
value stream coordination, 343–346
web address, 337
Value stream level backlogs
abstract, 193
in the Big Picture, 193–197
capacity allocation, 195–196
definition, 516, 521
details, 193–197
Little's Law, 197
PI planning, 195
pre- and post-PI planning meetings, 386
prioritizing, 194
queues, 197
refining, 194
solution integrity, optimizing, 195–196
value, optimizing, 195–196
wait times, 197

Value stream level epics, 488
Value stream objectives, creating, 230–231
Value Stream Performance Metrics, 314
Value stream PI objectives, definition, 521
Value Stream Predictability Measure, 313–314
Value streams
abstract, 473
in the Big Picture, 473–482
coordinating, 481
crossing boundaries, 477
definition template, 476
details, 473–482
development, 474–475, 477, 478–480
identifying, 475–478
operational, 474–475
portfolio level, 473–482
reducing time to market, 481–482
triggering value flow, 474
types of, 474–475
value stream mapping, 481–482
Value streams, ARTs
coordinating, 481
multi-ART value streams, 480
multiple value stream ARTs, 478–479
single value stream ARTs, 479–480
Value streams, portfolio level
connection to program level, 157
definition, 521
multi-ART, 211
size, organizing the ART, 162–163
splitting, 163–165
strategic themes, 425–426
Variability, assuming. See Assuming variability (SAFe Principle #3).
Variable solution intent, 354, 355–356
Velocity
derived, calculating, 104
stories, estimating, 139
workload, estimating, 114
Version control, DevOps, 277
Vision
abstract, 295
in the Big Picture, 295–300
definition, 521
details, 295–300
enterprise, 421
program, 299
roadmap view, 299–300
spanning palette, 295–300
strategic themes, 426
Vision, portfolio
capturing in solution intent, 298–299
characteristics of, 296–297
definition, 296
solution vision, 297–298
Visualize and limit WIP (SAFe Principle #6). See WIP (work in progress), visualizing and limiting.
Visualizing progress. See also Kanban.
BVIRs (big visual information radiators), 95
DSU (daily stand-up) meeting, 95, 121
iteration execution, 120–121
story boards, 95
VSE (Value Stream Engineer)
abstract, 167
in the Big Picture, 167–170
definition, 521
details, 167–170
program level, 167–170
reporting structure, 169
responsibilities, 167–168
as servant leaders, 169
value stream Kanban, 191
Wait time
program backlogs, 197
versus queue length, 62
value stream backlogs, 197
Wake, Bill, 137
Waterfall development
intra-iteration waterfalls, 122–123
mixing with ART (Agile Release Train), 158
phase-gate milestones, 59–60, 324
"Technical Strategies for Agile and Waterfall Interoperability at Scale,” 391
Web addresses, Big Picture
guidance, 489
portfolio level, 411
program level, 153
SAFe foundation, 1
SAFe principles, 43
spanning palette, 271
team level, 73
value stream level, 337
Websites, SAFe (Scaled Agile Framework)
Case Studies articles, 37
interactive guide to SAFe. See Big Picture.
"Mixing Agile and Waterfall Development,” 158
overview, 1
Scaled Agile SPC membership site, 39
Weighted Shortest Job First (WSJF). See WSJF (Weighted Shortest Job First).

WIP (work in progress)
  limits, Kanban, 100–101
  managing iteration execution, 121–122
  shedding excess, 231

WIP (work in progress), visualizing and limiting (SAFe Principle #6). See also Kanban.
  in the Big Picture, 61–62
  flow limit, 61

Wishful thinking, 492

Workflow management, program level, 156

WSJF (Weighted Shortest Job First)
  abstract, 183
  in the Big Picture, 183–186
  cost of delay, calculating, 184–185
  definition, 522
  details, 183–186
  job duration, 185–186
  job size as a proxy for duration, 186
  prioritizing jobs, 183–186
  program level, 183–186
  WSJF, calculating, 183–184