The Project Management Blueprint Part 2: A Comprehensive Comparison of Waterfall, DAD, SAFe, LeSS, and Scrum@Scale

DARREN HAGMAN
Darren is a veteran developer, scrum master, and project manager with deep experience in both waterfall and agile methodologies.

Overview

In Part 1 of the Project Management Blueprint we covered Lean Software Development, Agile, Scrum, and Kanban software development methodologies and how they all trace their roots back to Lean Manufacturing. These methodologies are usually deployed on a single team level. However, complexity grows as projects and project teams become bigger and new approaches are needed to be agile at scale.

In Part 2, we will first dive into how project managers use the waterfall methodology, which is the most common framework for software development at traditional

companies. Juxtaposed to that, we will cover the most popular frameworks that try to incorporate agile principles at scale–Disciplined Agile Delivery (DAD), Scaled Agile Framework (SAFe), Large Scale Scrum (LeSS), and Scrum@Scale.

Waterfall

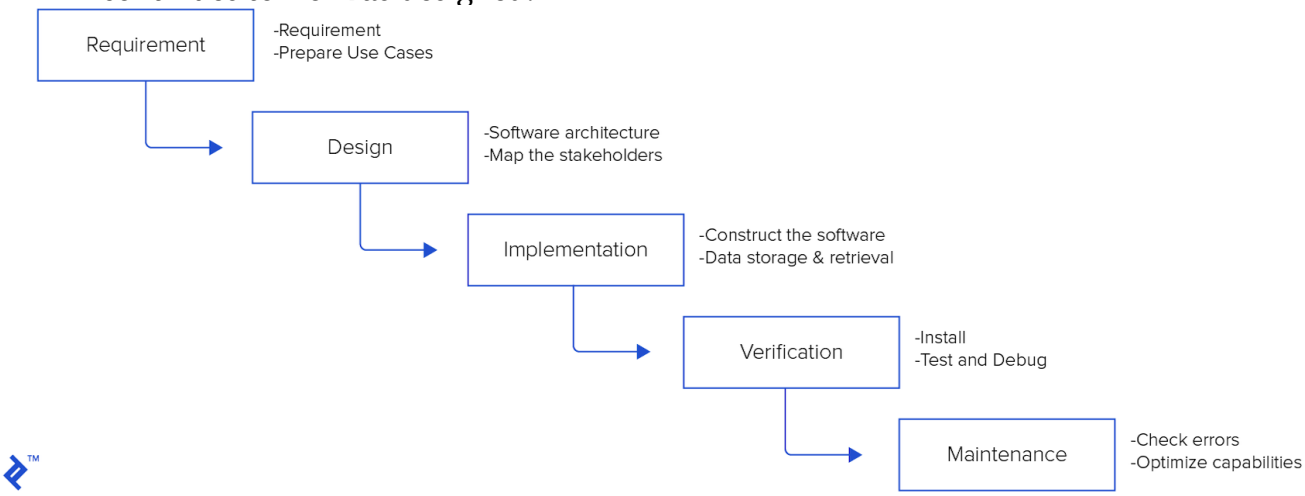
The waterfall methodology (also known as the software development life cycle model (SDLC)) is a more traditional methodology where software development cascades from one phase to the next like a waterfall. The phases do not overlap and have specific entrance and exit criteria for moving from one phase to the next.

The six life cycle stages of the original waterfall model are:

1. **Requirements:** In this phase, the expectations and goals of the project are defined, and requirements are analyzed and documented extensively, usually by a business analyst. The requirements are reviewed and approved before exiting this phase.
2. **Design:** After the requirements are approved, work commences on architecting and designing the product to meet the approved requirements. The physical architecture, component architecture, database design, detailed component, module design, and other aspects of design are documented by a software architect or designer. It is then reviewed and approved before beginning implementation.
3. **Implementation:** After the design is approved, implementation or coding of the software according to the requirements and design is done by software developers.
4. **Verification:** After the implementation is complete, the software is tested by the testing or QA team to ensure that the requirements and design are met and that

the desired level of quality is achieved. Defects are found, logged, triaged, and in many cases, fixed.

5. **Release and maintenance:** After testing and debugging are completed, the product is released to the client and installed. Often, a round of testing happens to ensure that the installation was successful. After the product is delivered, ongoing maintenance and support take place to ensure that the product continues to work as designed.



Advantages of Waterfall

There are some advantages to waterfall and it is suitable for certain types of projects, but there are also some serious disadvantages. Waterfall is best suited to shorter projects where the requirements, technology are well understood and are not likely to change in any significant way.

If applied to the right type of project, some of the advantages of the waterfall model:

- **Simplicity:** Waterfall is simple to implement due to identifying the scope up front and due to the rigid phases and a clear transition from one phase to the next.

- **Visibility:** Progress is more easily measured and seen by stakeholders as the full scope of the work is known in advance and as the project transitions from one stage to the next.
- **Documentation:** Scope, requirements, and plans can be thoroughly thought through and well documented, which makes it easier for less experienced teams to work on the project.
- **Phased work:** Due to the rigid roles and transition between phases, it is possible for project resources to work on other projects when their primary phase isn't in progress.

Disadvantages of Waterfall

Waterfall is not suited for longer projects where the requirements are not well understood and/or likely to change and/or where there is significant technical risk. In today's age where market conditions are constantly changing and time to market is critical, this applies to most software projects.

Disadvantages of the waterfall model, which mostly center around its inability to adapt to change, include:

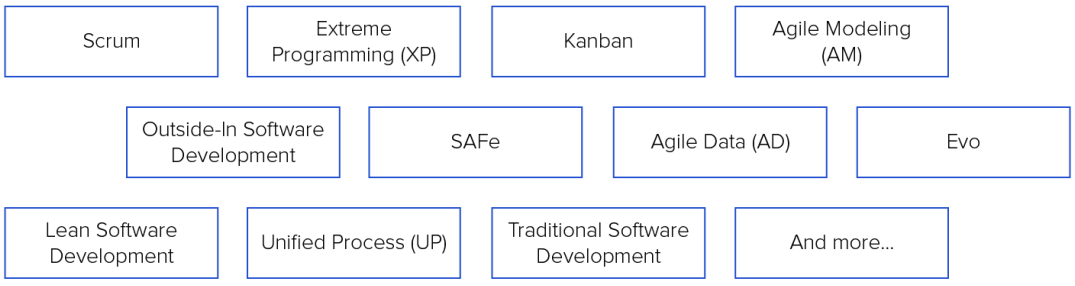
- **Monolithic scope:** It rewards stakeholders to think of EVERYTHING when defining the scope of the project, leading to a monolithic scope.
- **Late client feedback:** It is hard for stakeholders and especially clients to imagine the full detailed scope of a project. Since waterfall exposes clients to project results mostly in the last stages of the project, then inevitably it becomes hard to incorporate client feedback into the project

- **Requirements change:** In longer projects market conditions, and therefore project goals and requirements, are at very high risk of changing during the project.
- **Value created at the end:** Waterfall requires a lot of the work up front, meaning that value is not produced until late in the project.
- **Phase interdependency:** Incorporating changes often results in requirements and/or design rework which may impact other areas of the project. The dependency of later stages upon earlier stages can make small changes in the project disproportionately challenging.

Disciplined Agile Delivery (DAD)

Disciplined agile delivery (DAD) was formalized by Scott Ambler at IBM and Mark Lines and expands upon the agile and scrum frameworks, recognizing that non-agile parts of an organization are usually involved in some capacity in delivering a software project. This framework explicitly includes activities from IT operations, enterprise architecture, portfolio management, finance, and procurement into the full delivery lifecycle. DAD aims to increase overall business agility in a pragmatic way.

Disciplined Agile Delivery (DAD)



Source: Disciplined Agile Consortium

Main Principles and Components

ROLES

DAD has considerably more roles than scrum and is broken down into two categories of team roles. Primary roles are filled by people who work with the project on a constant basis. Secondary roles are typically introduced temporarily to help the team with scaling or other issues. DAD has these additional roles because it addresses the entire solution delivery lifecycle and because it recognizes the various types of needed temporary and supporting roles found in the real world

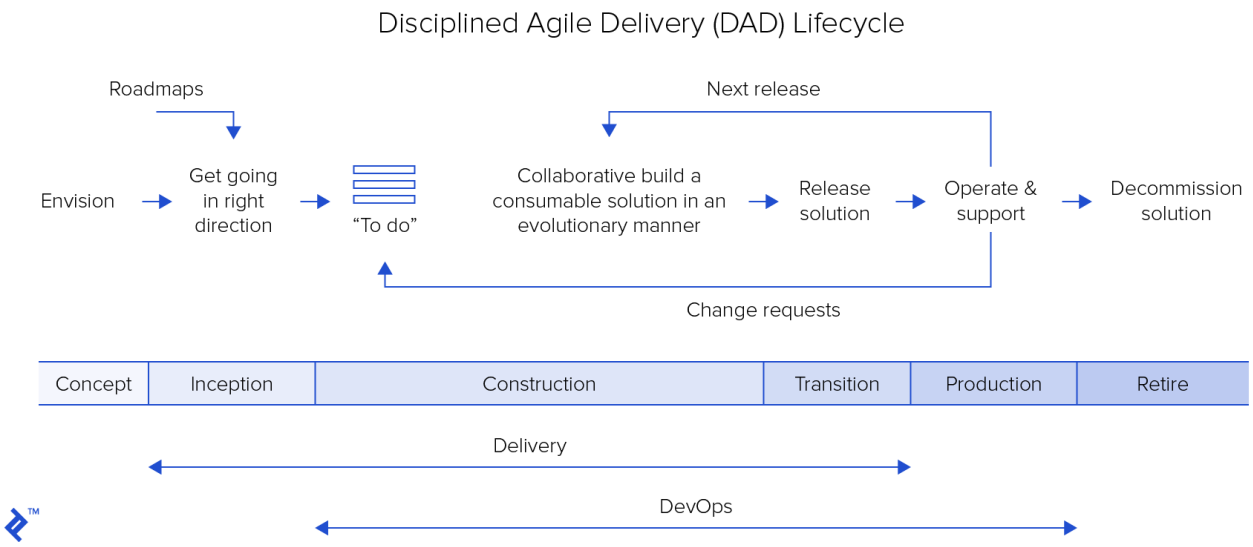
Primary roles:

1. **Stakeholder:** Someone who depends on your team finishing the project: client, end-user, or internal colleague.
2. **Team member:** People within the team that actually do the planned work: developers, designers, testers, etc.
3. **Team lead:** Analogous to the scrum master, the team lead works as a servant-leader for the team by removing impediments, facilitating team cohesion and spreading agile values.
4. **Product owner:** Sometimes referred to as the “voice of the customer.” The product owner represents the stakeholders and maintains the prioritized list of work that needs to be done.
5. **Architecture owner:** Responsible for mitigating architecture risk at scale. This role is typically filled by a senior developer within the team as it requires a deep technical background and solid business domain knowledge.

Secondary roles:

1. **Specialist:** People who join the team temporarily to help out in a specialized role. For example, a data analyst may join to provide research capabilities in the early stages of a project.
2. **Domain expert:** Tax consultants, legal advisers, and other people who have domain expertise and help out the team on related challenges.
3. **Technical expert:** Database administrator, security expert build master, etc. These people help out the more generalized team members at key points in the life cycle.
4. **Independent tester:** While testers are usually part of the main team, in some cases life regulatory requirements or very complex systems, independent testers work in parallel to validate deliver work.
5. **Integrator:** At scale, different teams are working on different parts of the whole system. An integrator helps the team integrate their part with the whole system and manages dependencies.

LIFECYCLE SUPPORT



DAD promotes a full delivery lifecycle, not just the programming and release part covered by agile/scrum, but also the inception phase where the project vision is defined

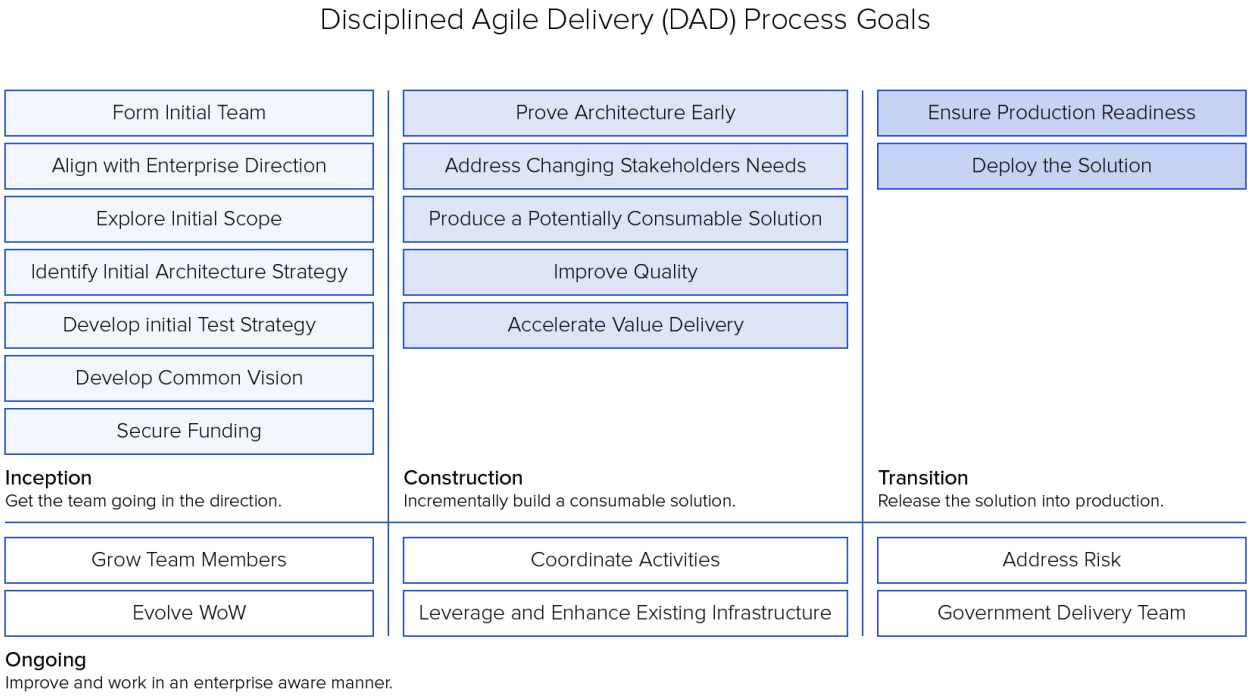
and approved and the support and retirement phases after release. Currently, DAD supports 6 different lifecycles:

- The Agile Lifecycle: A Scrum-based Project Lifecycle
- The Lean Lifecycle: A Kanban-based Project Lifecycle
- The Continuous Delivery: Agile Lifecycle
- The Continuous Delivery: Lean Lifecycle
- The Exploratory (Lean Startup) Lifecycle
- The Program Lifecycle for a Team of Teams

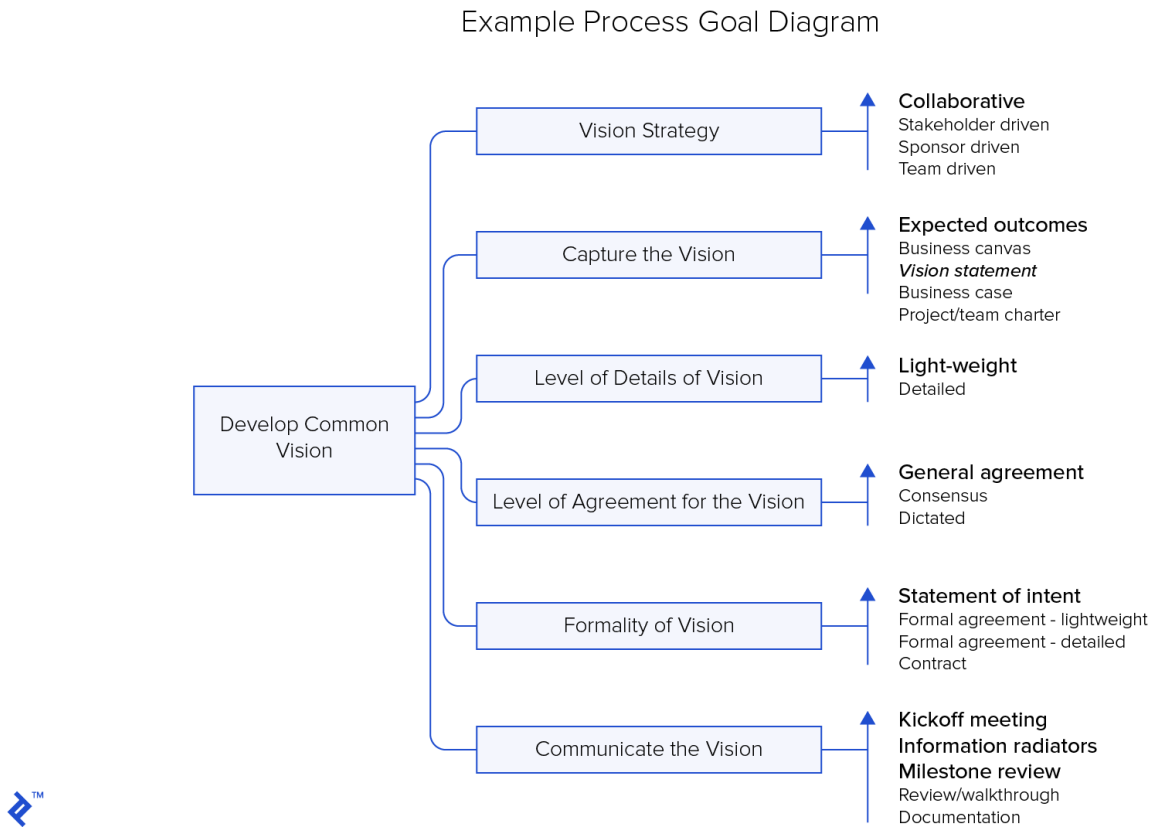
These lifecycles account for different work styles, levels of company agility, and other situations that the teams might find themselves in. The main point is that these lifecycles act as suggestions. DAD promotes pragmatism over purism since every situation is unique and disciplined agile practitioners should adopt the agile process to their needs.

PROCESS GOALS

DAD uses a goal-driven approach to creating and adapting agile processes. The authors of this methodology outline 21 most important and common processes that most teams will face during their life cycles.



All of these processes have documented decision points that will require the team to decide how they will structure that process. Each decision point provides suggested techniques or practices that can be used to implement the decision. You can see an example of this in the image below. A process “Develop Common Vision” has 6 decisions that should be made. Each of those decisions has 2 to 5 suggested practices. The arrows indicate that DAD authors have ordered the list with the top item being the best practice and the bottom item being the worst practice in this list. The_ **bolded italic**_ text signifies good starting points for new teams, who are just starting out with DAD.



SCALING OF DAD

Disciplined Agile Delivery approaches scaling from two different angles:

- Tactical agility at scale
- Strategic agility at scale

Tactical agility tries to address individual team scaling factors such as size, geographic distribution, project complexity, etc, through situational application of the process goals and their suggested practices.

Strategic agility tries to address scaling thru the application of agile and lean strategies broadly across the entire organization by expanding the framework to address different areas of the organization:

- Disciplined DevOps: covers using DevOps to provide more effective outcomes to an organization.
- Disciplined Agile IT (DAIT): covers how to apply agile and lean strategies to all aspects of IT.
- Disciplined Agile Enterprise:. covers how to apply lean and agile throughout an enterprise.

SAFe

Scaled Agile Framework (SAFe) is the most popular as well as the most complex and comprehensive scaled Agile framework right now. 29% of respondents in the Annual State of Agile Report claim that they use this framework in their organizations. In turn, there are many SAFe project managers in the market.
The inception of SAFe was Dean Leffingwell's book "Scaling Software Agility: Best Practices for Large Enterprises," published in 2007. Leffingwell is now the chief methodologist of SAFe, but many other people also contribute to this framework. Currently, in version 4.6, this framework resembles a software product with versioning, backward compatibility, and various components.

Main Principles and Components

The primary goal of SAFe is to facilitate the creation and growth of a Lean Enterprise as it recognizes that many different types of companies are, in part, software companies that need to continually deliver value in the shortest, sustainable time period.

SAFe for Lean Enterprises tries to create a Lean Enterprise by providing a knowledge base of proven principles, competencies, and best practices.

SAFe 4.6 defines Five Core Competencies of the Lean Enterprise. Each competency is a set of related knowledge, skills, and behaviors, which together enable organizations to excel:

1. **Lean-Agile leadership**: Describes how leaders drive and sustain organizational change through learning, teaching, and implementing SAFe's Lean-Agile mindset.
2. **Team and technical agility**: Describes the skills, principles, and practices that are needed to create high-performing Agile teams.
3. **DevOps and release on demand**: Describes how implementing DevOps and a continuous delivery pipeline provides organizations with the capability to release product increments at any time necessary to meet demand.
4. **Business solutions and lean systems engineering**: Describes how to apply lean-agile principles and practices to the specification, development, deployment, and evolution of large, complex software applications
5. **Lean portfolio management**: Aligns strategy and execution by applying lean and systems thinking approaches to strategy and investment funding, agile portfolio operations, and governance.

Each of the core competencies map directly to their respective level in the SAFe process diagram except Lean-Agile Leadership which encompasses the entire process.

LEAN-AGILE LEADERSHIP COMPETENCY

The primary goal of the Lean-Agile Leadership Competency is to help transform the organization to a lean-agile enterprise. This is done by learning, practicing, and teaching SAFe's Lean-Agile mindset, values, principles, and practices.

SAFe's Core Values guide the transformation to the lean enterprise. At every opportunity, a leader's behavior plays a critical role in promoting them. The core values are:

1. **Alignment**: Communicate the mission, portfolio strategy, and solution vision. Conduct relevant briefings, and participate in program increment (PI) planning and backlog maintenance.
2. **Transparency**: Visualize all relevant work.
3. **Built-in quality**: Engage in practices to deliver quality throughout the life cycle. Refuse to accept low-quality work. Support investments in maintenance and reducing technical debt.
4. **Program execution**: Participate as business owners in PI execution and establish business value. Ensure that the scope is aligned with demand and capacity. Aggressively remove impediments and demotivators.

SAFe core values are supported by embracing the Lean-Agile Mindset and applying SAFe Principles:

1. Take an economic view
2. Apply systems thinking
3. Assume variability; preserve options
4. Build incrementally with fast, integrated learning cycles
5. Base milestones on an 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

These principles are similar to Lean and Agile principles. Finally, transforming the organization is accomplished by following the SAFe Implementation Roadmap.

TEAM AND TECHNICAL AGILITY COMPETENCY/TEAM LEVEL

The Team and Technical Agility Competency describes the skills, principles, and practices needed to create high-performing agile teams who create high-quality solutions. Two key characteristics are critical:

- **Team agility**: teams adopt Agile practices and principles, which enable them to work, learn, and adapt on a reliable cadence
- **Technical agility**: teams apply Agile technical practices that ensure the code and component quality, and maintainability of the code they produce\ Quality is a big focus in the team and technical agility competency. To achieve this, agile engineering techniques such as Behavior Driven Development (BDD) and Test-driven development (TDD) are applied to increase quality and flow. Fast flow depends on building quality throughout the system as errors can severely impact flow and delay releases.

The Team Level of the SAFe diagram describes how individual Agile teams operate. All teams are part of the Agile Release Train which work towards delivering a Product Increment. Most of the traditional agile/scrum flow applies, where teams work in

iterations to deliver working systems. The roles of scrum master, product owner, and team member are used in SAFe as are most of the scrum activities and artifacts. Teams are also supported by program level roles such as Product Management, System Architect, and other shared services. Kanban is used to help visualize the flow of features through the delivery lifecycle and interactions and handoffs between teams.

DEVOPS AND RELEASE ON DEMAND COMPETENCY/PROGRAM LEVEL

The DevOps and Release on Demand Competency describe how "implementing DevOps and a continuous delivery pipeline provides the enterprise with the capability to release value, in whole or in part, at any time necessary to meet market and customer demand." DevOps works to align development, operations, the business, and other areas to work together to deliver business results. While not every organization needs to release as often as some of the industry leaders of the DevOps movement (Amazon releases every few seconds), all organizations need to be able to release on demand.

- DevOps provides the culture, automation, lean-flow, measurement, and recovery (CALMR) approach that enables continuous delivery and release on demand

- Agile release trains (ARTs) are teams of agile teams that are organized to deliver value on demand via a continuous delivery pipeline

The Program Level of the diagram describes the roles and activities needed to continuously deliver via an **Agile Release Train (ART)**. This level works in a similar iterative way to the team level but integrates multiple agile teams and includes more cycles. The ART is an agile team of teams comprised of 5 to 12 teams (50 to 125 people) including the traditional agile roles as well as critical program roles like **Release Train Engineer (RTE)** and Product Management. The ART delivers in 8-12 week **Program**

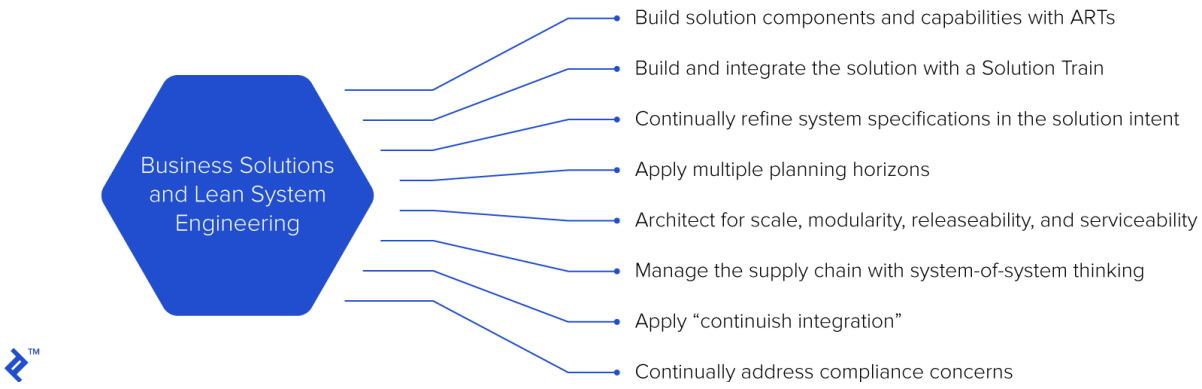
Increments (PI) which are planned via **PI Planning** and led by a **Product Manager**.

The progress of PI features, epics, etc is tracked and managed via a Program Kanban board. The RTE acts as the Scrum Master on the ART. Daily synchronization meetings include team Daily Standups, Scrum-of-Scrums (RTE & Scrum Masters), PO Sync (Product Management & Product Owners), and ART Sync (Scrum-of-Scrums and PO Sync together). Each PI has a System Demo and a Retrospective.

BUSINESS SOLUTIONS AND LEAN SYSTEMS ENGINEERING COMPETENCY/LARGE SOLUTION LEVEL

The Business Solutions and Lean Systems Engineering Competency describes "how to apply Lean-Agile principles and practices to the specification, development, deployment, and evolution of large, complex software applications and cyber-physical systems".

In addition to the SAFe principles, applying the following 8 principles when working on large solutions are key to this competency:

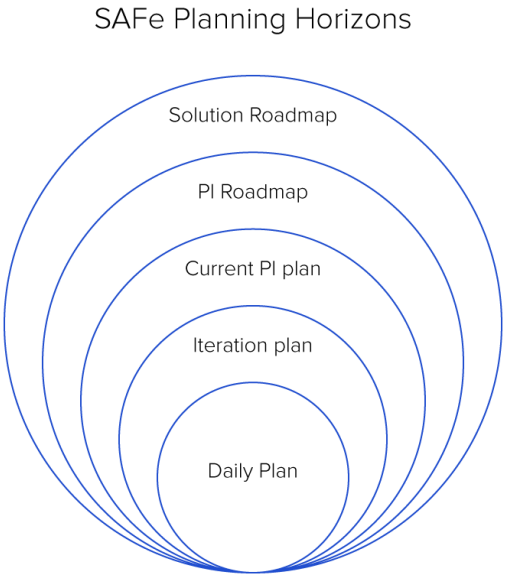


The Large Solution Level contains the roles, artifacts, and processes needed to build large and complex solutions. Multiple ARTs are working together on **Solution Trains** to deliver **Solutions**. The primary objectives are to:

- Manage frequent integration

- Continuously address compliance concerns

- Architect for scale, modularity, releasability, and serviceability



Solution Management controls the content of a **Solution and the Solution Train Engineer (STE)** guides the work. **Solution Architect** is responsible for maintaining good architecture for all the ARTs in the Solution. **Pre and Post PI Planning** is used to plan Solutions delivered via multiple Program Increments. A **Solution Backlog** contains **Capabilities** and **Solution Epics** and is tracked via a **Solution Kanban** board

PORTFOLIO LEVEL/LEAN PORTFOLIO MANAGEMENT COMPETENCY

The Lean Portfolio Management Competency "aligns strategy and execution by applying Lean and systems thinking approaches to strategy and investment funding, agile portfolio operations, and governance."
Lean Portfolio Management focuses on the following areas:

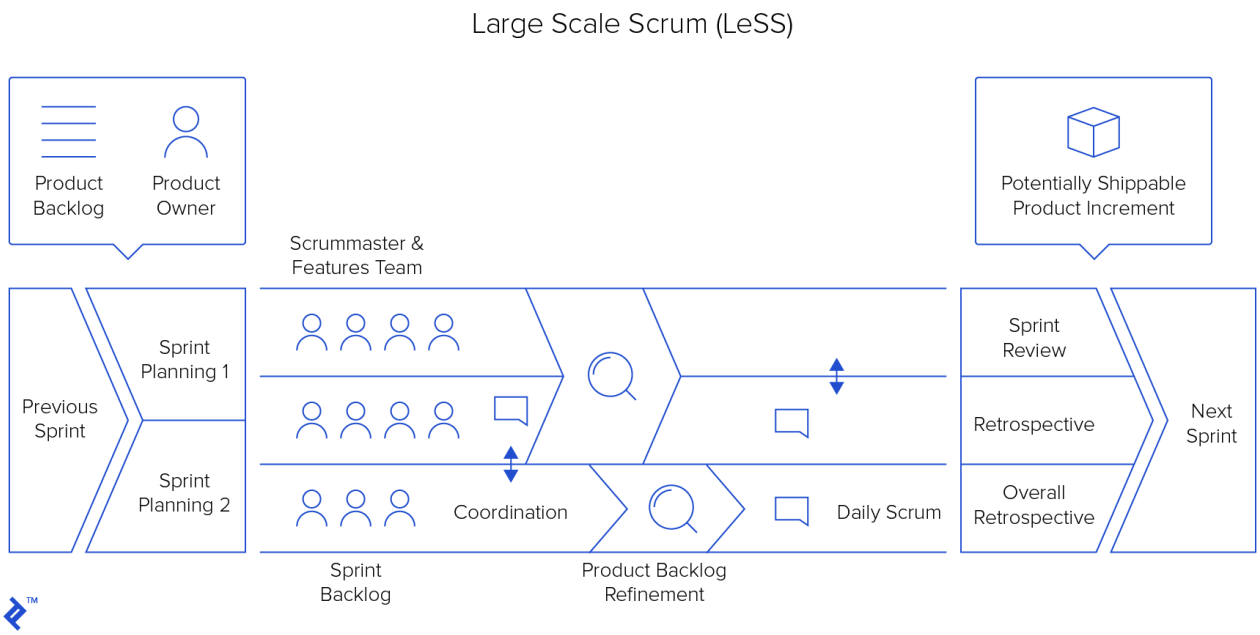
1. Strategy and investment funding: connects the portfolio to enterprise strategy, funds value streams, and establishes portfolio flow

2. Agile portfolio operations: coordinates the value streams, support program execution, and operational excellence

3. Lean governance: forecasts budgets, measures portfolio performance and enforces compliance

The Portfolio Level contains the principles, practices, and roles needed to initiate and govern a set of development Value Streams. A **Portfolio Backlog** contains **Business Epics** and **Enabler Epics** and is tracked via a **Portfolio Kanban* board. **Lean Portfolio Management (LPM)** decides on what value streams are in a portfolio and includes the highest decision-makers in an enterprise. An **Enterprise Architect** guides the work and coordinates across Value Streams.

LeSS



[Large scale scrum (LeSS)] framework was created by Craig Larman and Bas Vodde, who have based it on their experience in the financial and telecommunication industries. As the name implies, LeSS promotes having as few processes and procedures as possible to have many Scrum teams work together. Scaling is hard because people make it complex, so the goal here is to make it as simple as possible.

Main Principles and Components

"LeSS is Scrum, applied to many teams, working together, on one product". LeSS is based on ten principles which will seem familiar to anyone who is familiar with Lean-Agile principles:

1. Large-Scale Scrum is Scrum

2. Empirical process control

3. Transparency
4. More with less
5. Whole-product focus
6. Customer-centric
7. Continuous improvement towards perfection
8. Systems thinking
9. Lean thinking
10. Queuing theory

LeSS has only two main roles, both of which are borrowed from Scrum:

1. **Product owner:** Works with 2-8 teams.
2. **Scrum master:** Works with 1-3 teams.

All the teams work with the same **product backlog** in 1-4 week sprints. The teams work in parallel, meaning that they start and end sprints at the same time. At the end of the sprint, the teams collectively deliver a **product increment**. It might seem nearly impossible for one product owner to work with 8 teams. LeSS promotes moving the responsibility of product backlog item clarification to the teams. In turn, the teams must be cross-functional and contain not only coding, design, and testing competencies but also business domain knowledge. More importantly, the teams have to be empowered to be able to reach out to customers.

SPRINT PLANNING

Planning is split into two parts:

1. **Sprint planning 1:** Where representatives of teams meet with the product owner and decide on which backlog items they will take on and discuss any potential cooperation that might be needed between the teams during the sprint.
2. **Sprint planning 2:** Same as in traditional scrum, each team gathers separately to create a plan for how the backlog items will be done.

PRODUCT BACKLOG REFINEMENT

Product backlog refinement (PBR) is done during the sprint to prepare product backlog items for sprint planning. LeSS does not offer rules how to do PBR and leaves it up to the companies to figure out their most effective process themselves. PBR involves three key activities:

1. Splitting big items.
2. Detailing items until ready.
3. Estimating.

END OF SPRINT

At the end of each sprint three things happen:

1. **Sprint Review:** A shared Sprint demo, where teams and customers explore what was done during the Sprint and what should be done next.

2. **Retrospective:** Each team holds their own retrospective to improve their process.
3. **Overall retrospective:** Teams, product owners, and scrum masters get together to inspect and adapt organizational practices to be more effective.

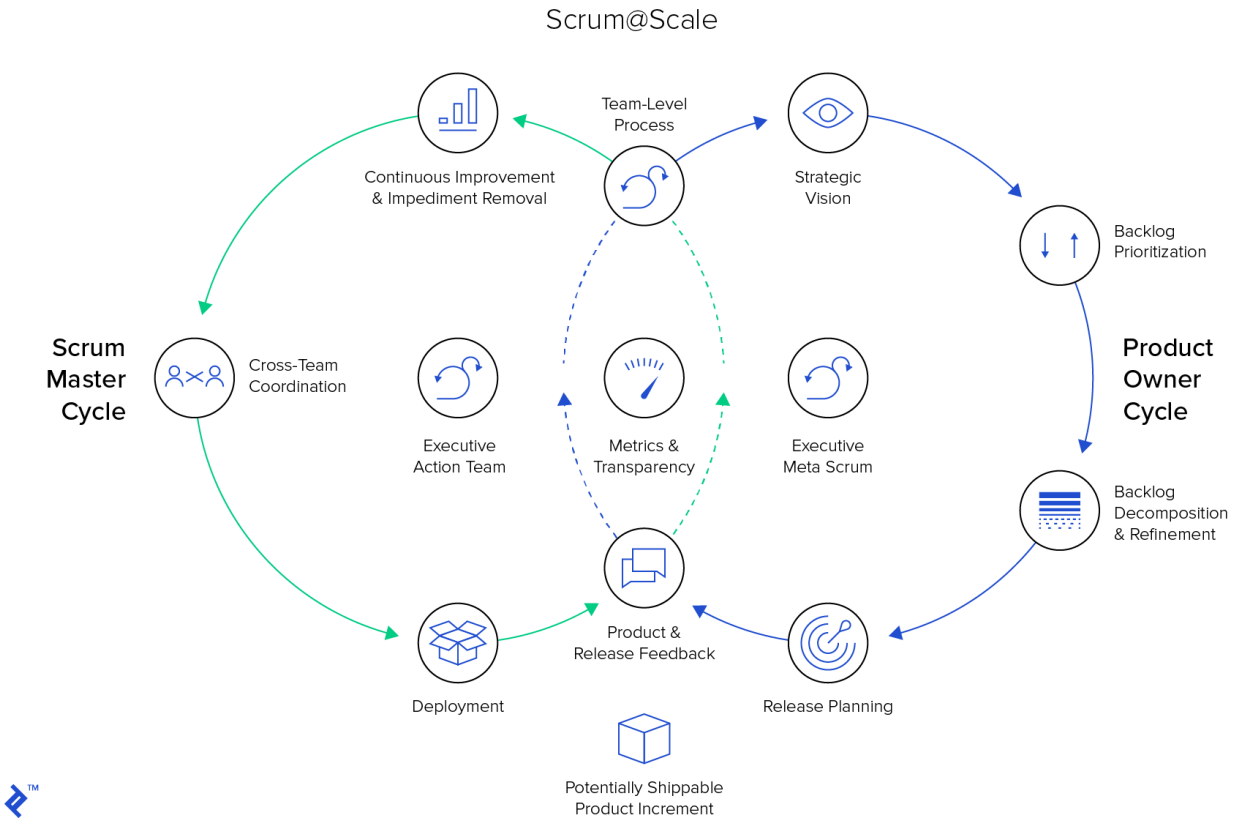
Scrum@Scale

Scrum at Scale and Scrum@Scale are used interchangeably. This methodology was introduced by Jeff Sutherland in 2014, who created the Scrum methodology and was one of the signatories of the Agile Manifesto.
Scrum@Scale takes Scrum as its starting point and offers a simple, lightweight framework to scale Scrum with a "minimum viable bureaucracy". It is less prescriptive than the others scaled agile methodologies and can be considered as a meta-framework. It highlights the scaling issues and areas and offers a mental framework for how they could be addressed.

Main Principles and Components

Scrum@Scale is a framework that radically simplifies scaling by using Scrum to scale Scrum. In Scrum, the "what" (product owner) is clearly separated from the "how" (scrum master). The same strategy is used in Scrum@Scale so that jurisdiction and accountability are well understood, eliminating waste and conflict.

Scrum@Scale contains two cycles to clearly separate jurisdictions: the **Scrum Master cycle** and the **Product Owner cycle** with two touchpoints: Team Level Process and Product Release Feedback.

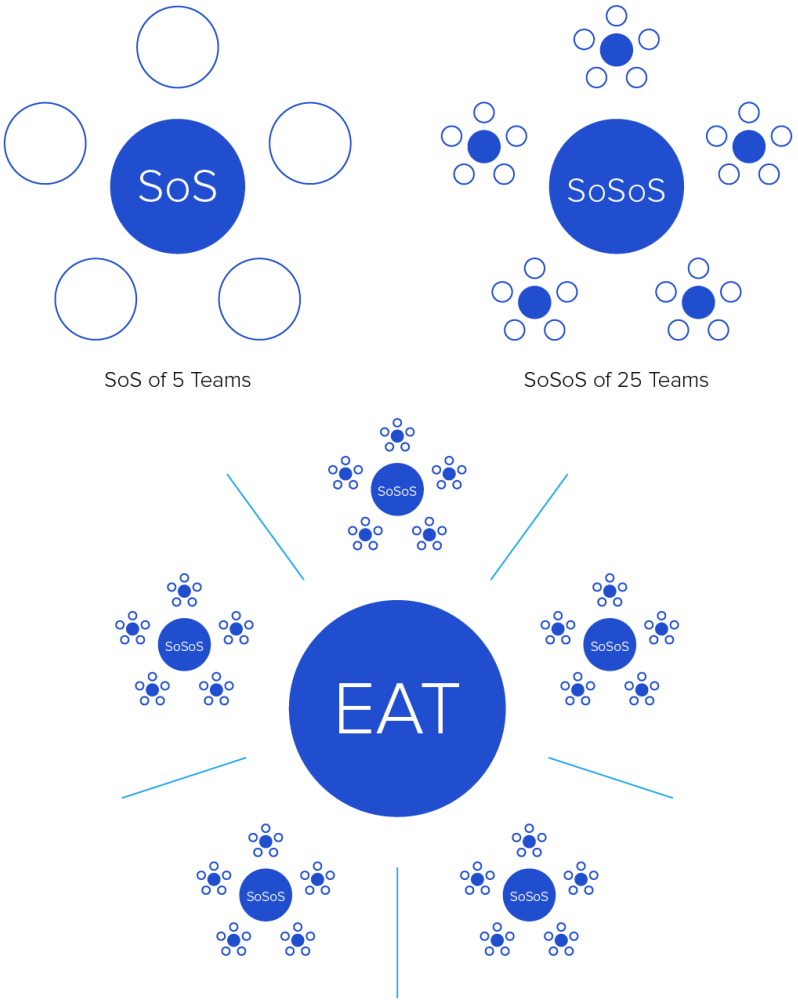


SCRUM MASTER CYCLE

The **scrum master cycle** is responsible for how the things that the product owner cycle identified will be built. In Scrum@Scale, individual Scrum teams have the same roles, artifacts, activities, and ceremonies as traditional Scrum.

Scrum teams are grouped into a **Scrum of Scrums (SoS)** which jointly responsible for producing a joint product increment. They participate in joint backlog grooming and prioritization, hold retrospectives, maintain impediment backlogs, and they hold a **Scaled Daily Scrum (SDS)** (similar in format to the daily scrum) to coordinate the teams and remove roadblocks. The SDS is attended by at least one representative (usually the team's scrum master) of each of the participating teams and is led by the **Scrum of Scrums master (SoSM)** who is responsible for coordinating with the scrum teams and the product owners.

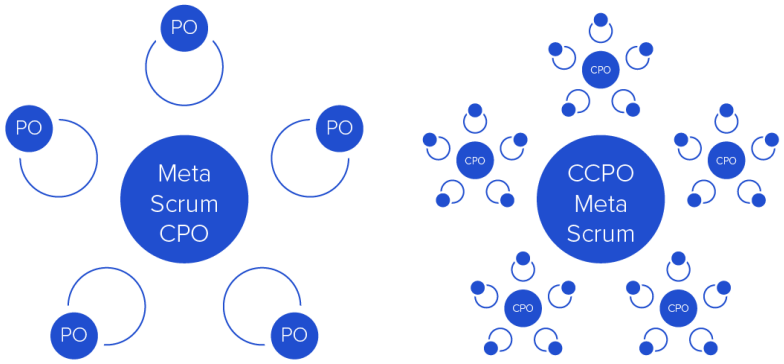
If further scaling is needed, there is a **scrum of scrum of scrums (SoSoS)** created out of multiple SoS which would also meet daily, and so on. The overall leader is the **Executive Action Team (EAT)** which is responsible for promoting Agile in the organization, coordinating Scrum teams as needed, and for being the final stop for removing impediments.



PRODUCT OWNER CYCLE

The **Product Owner Cycle** is responsible for what product or service will be created and all the activities needed to support that. Product Owners are assigned to Scrum teams and carry out all the activities of their role as defined in Scrum. Product owners are grouped into **Product Owner Teams** which map to the SoS teams. Product

Owner Teams meet daily at a **Meta Scrum** to discuss a high-level strategy for the teams, and coordinate as needed with the corresponding SoSM and other product owners and stakeholders.. Meta Scrums are led by a **Chief Product Owner (CPO)**. Product Owners scale in a similar way to the scrum master cycle, depending on the size of the organization and culminates in an **Executive Meta Scrum (EMT)**, which is responsible for company-wide priority setting.



SoS of 5 Teams

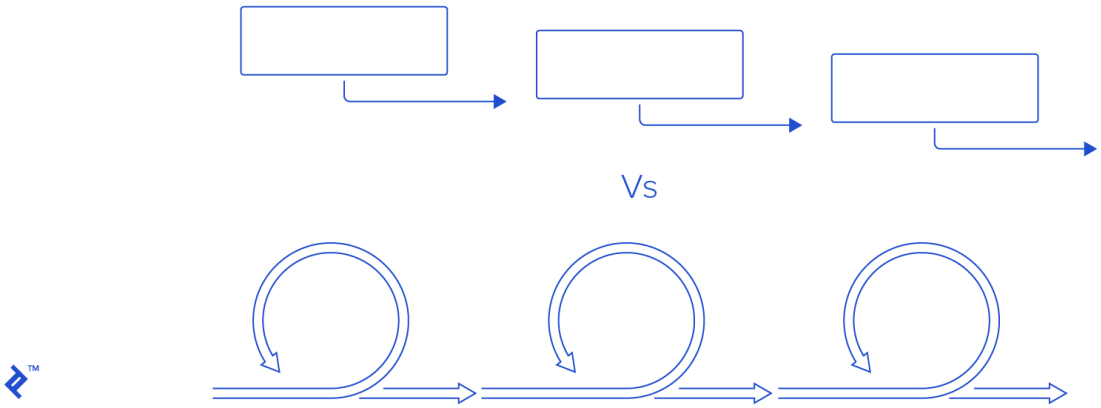
MetaScrum of 25 Teams

IMPLEMENTING SCRUM@SCALE

Implementation of Scrum@Scale starts with creating a scalable reference model, i.e. a small set of teams using scrum at a small scale. This is done to resolve any organizational policies and development practices that hinder agile. Scrum@Scale suggests resolving these early because all teams are likely to face these organization wide-problems and the consequent frustrations could hinder the adoption of agile. The reference model is then used as a pattern for scaling scrum to other teams and departments.

Executive action team (EAT) has to be created initially to implement the reference model. EAT is comprised of individuals who are politically and financially empowered within the organization, as they will be able to implement the organizational policy changes.

Conclusion



In this second part of the project management blueprint, we've covered some of the most popular frameworks used on larger projects or programs. Waterfall is still widely used in many organizations, and while it has many disadvantages due to its inflexible processes, it sometimes makes sense to use this framework when the projects are small and the scope is well-defined and unlikely to change.

As companies encounter larger, more complex projects with constantly changing requirements or goals, they look to more Agile approaches. As Agile was originally meant for small teams of 5-9 people, various Agile practitioners have come with multiple ways to scale agile from single teams, to multiple teams, to the entire enterprise. In this article, we covered Disciplined Agile Delivery (DAD), Scaled Agile Framework (SAFe), Large Scale Scrum (LeSS), and Scrum@Scale.

In the final part of the project management blueprint, we will cover a few project management specific frameworks like PMP (PMBOK) and PRINCE2. We will also go over some innovation processes and frameworks like "jobs to be done" (JTBD) and "design thinking."

UNDERSTANDING THE BASICS

What is the difference between waterfall and agile methodologies?

Waterfall methodology involves a lot of up-front planning and the results are delivered at the end of a project. Agile methodology promotes lightweight planning and multiple iterations to deliver results in small increments.

Can waterfall and agile be used together?

Yes. Dividing a waterfall project into parts, where some of the work would be done iteratively in sprints would be an example of the two methodologies used together. Hybrid methodologies also help to combine both approaches to get the most benefits.

What is a hybrid methodology?

TAGS

AgileWaterfallScrum@ScaleLeSSSAFeDAD

Darren Hagman
Project Manager

ABOUT THE AUTHOR

Darren is a veteran developer, scrum master, and project manager with deep experience in both Waterfall and Agile methodologies from working in the transportation, archiving, eCommerce, and aerospace industries. Darren spent ten years working in developer and lead/architect roles before becoming a project manager. Darren understands the challenges of implementing Agile practices in a team and how to effectively interface with the Waterfall world.