The Home of Agile

Contents

Purpose 2
Why Agile? 2
Organizational Preparedness 3
Agile Fundamentals 5
Connection to Lean 14
Connection to DevSecOps 16
Structuring Work Around Products and Value 17
Agile Planning 20
Common Roles/Responsibilities 23
Common Tools 24
Measuring Progress and Success 25
Common Agile Frameworks 26
How to Get Started – An Example Using Scrum 28
Scrum Ceremonies 29
Let's Get Started! 31
Key Terms 32

Purpose

The purpose of this document is to provide an overview of Agile values, principles, frameworks, roles, and vocabulary to provide an understanding of the value generated from Agility and how it differs from traditional/predictive/waterfall project management.

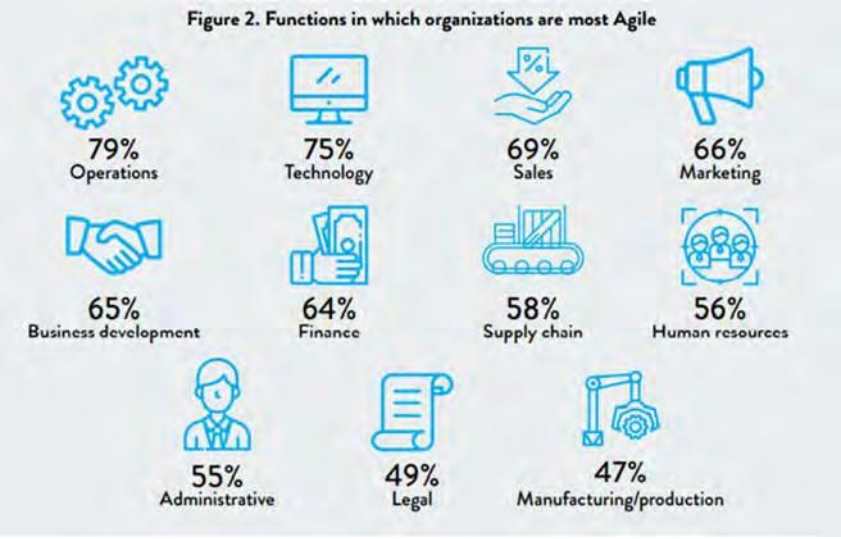
This document provides foundational information the reader can use to gain a broad understanding of overarching Agile concepts and how Agile relates to Lean and DevSecOps practices.

Why Agile?

Innovation and product lifecycles, resources to deliver value, and competitor response times are all shrinking. Organizations are asked to deliver more with less, deliver better products/services faster, and adapt to customer needs and competitor capabilities at a faster pace.

In a 2023 survey of 1000 C-Suite executives by Forbes Insights/Scrum Alliance, 81% of respondents considered Agility the most important characteristic of a successful organization and believed it produced significant benefits to the organization (as in Figure 1 below).

Further, Agile was adopted across the organizations in many areas outside of software development and technology (see Figure 2 below).



Agility provides organizations the ability to invest in valuable products and teams that perform with greater insight and knowledge, generating investments with less risk.

Agile focuses on smaller, more iterative, and incremental releases results in faster delivery of value and the ability to enhance value.

Investment Consideration	Waterfall	Agile
Knowledge of the Initial Investment	***Attempts to make large, long-term investments with limited information on Day 0.*** False confidence fixes scope used to drive cost and schedule estimates.	***Recognizes we don't know enough to make large, long-term investments on Day 0 but will learn over time.*** We don't know enough to fix scope and will need to adapt if we want to maximize value.
Payback Period	***Long-term, big-bang.*** Waiting for the promise of value years into the future and hoping what is delivered is considered valuable by stakeholders.	***Short-term, incremental returns that add up over time.*** Delivery of value occurs near-term and incrementally, delivering verifiable value to stakeholders over time.
Return on Investment	***Limited ability to improve/enhance return.*** Value demonstrated at the end of the project, with limited feedback loops to enhance value (stakeholders engaged at the beginning and end). Efforts to "control scope" minimize opportunities to enhance future value.	***Designed to constantly improve/enhance return.*** Value is demonstrated/delivered often to stakeholders, creating fast feedback loops that drive future direction. Scope adapts to enhance/maximize future value.
Future Investment Decisions	***Compliance with plan drives future investment decisions.*** Ability to deliver stated requirements at cost and schedule estimates is rewarded (with limited to no assessment of value).	***Value delivered and team performance drive future investment decisions***. Investment made in most valuable products are teams delivering the most value. Investment shifts away from low-value products and teams that cannot deliver.

Table 1: Agile Investment Considerations

Organizational Preparedness

Although the transition to Agile can generate significant benefits, there are significant challenges to consider as well (see Figure 03 below).

Shifting investment focus from stand-alone projects that begin and towards continuous evolution of valuable products, reorganizing or matrixing resources alone value streams required to generate those products, and breaking down barriers (policies, processes, ways of working) that inhibit the flow of value is challenging.

However, many of the key challenges below are centered around people and culture. It is critical that organizations are prepared to tackle change management associated with Agile's impact on people, policies/processes, and systems/tools, but also the change Agility will bring to the organization's culture.

To be successful leadership needs to embrace Agility, empower champions and train is required for everyone. Leadership and teams need to be Agile and stay the course even when there are bumps in the road. Agile coaching both during and after the initial transformation may help to ensure stickiness.

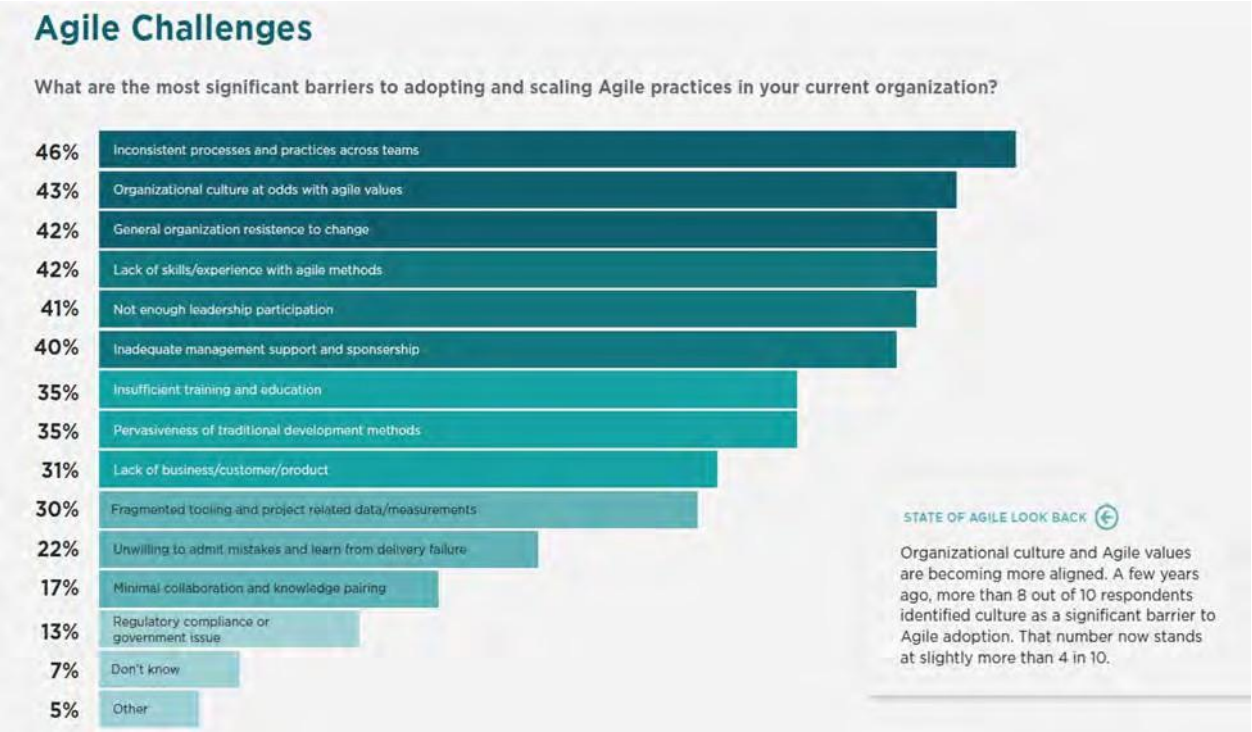


Figure 3: Agile Challenges

At the PMO level, a traditional program office in the government context is comprised of project and program managers, organized around projects/programs where the government acquires products developed by vendors that are typically delivered multiple years into the future.

Significant upfront effort occurs to determine all requirements, estimate cost/schedule, and develop a long-term plan for delivery results. This results in contract negotiation to (mostly) lock-in all requirements, schedule, and budget. The prime contractor does the work in a black box with the promise of a big value delivery in the future.

This approach creates two teams – one on the government side (primarily oversight) and one on the vendor side (developing the product). Since product isn't delivered/available until the very end of the project and requirements are (mostly) locked in, emphasis is placed on producing documentation to demonstrate progress and instill confidence.

Numerous designs, project schedules/IMS, metrics, risk/issue logs, etc., are generated that distract the team from delivering valuable product. Program officers spend a significant amount of their time reviewing these documents and providing feedback, guidance, direction. Since the program office can't assess the value of working product, success becomes managing vendors to deliver set requirements per the schedule at budget.

If corrective action becomes warranted by the government, it necessitates an all-hands-on-deck to create more documentation that further distracts the team from delivering value.

Lessons learned for future improvement are captured after the big-bang delivery multiple years in the future.

In contrast, an Agile program office shifts emphasis to helping to identify good investments - what products are delivering the most value and what teams are delivering the most value. This shifts the program office towards identifying the most efficient value delivery and away from how value is delivered (detailed tasks/activities).

Although program management may still be required, additional roles – the Product Owner and Scrum Masters are emphasized. Since Agile anticipates that requirements will change and stresses collaboration over contract negotiation, requirements are never locked in.

Instead, for each product that is deemed valuable, capabilities are targeted for delivery on an Agile Roadmap. Further, the government establishes a Product Owner that acts as the voice of the customer and has authority to routinely make decisions on what will be delivered and when that delivery will occur.

The Product Owner attempts to maximize value delivered taking into consideration emerging customer needs, new learning, and evolving competitor capabilities. Agile emphasizes delivering working product iteratively and incrementally (delivering along the shortest duration possible) and minimizing documentation/ overhead activities that distract the team from delivering value (if you can build the care for the customer to drive you don't need to waste time on a glossy pamphlet explaining how it feels to drive the car).

This necessitates vendors shift away from developing a product in a black box and big-bang delivering to the government. Instead, vendors provide software teams that are guided by the Product Owner and fully integrated with the government team (one team instead of two).

Continuous improvement occurs with each incremental delivery, which allows for more efficient delivery of value in the future.

Transparency provides the ability for the program office to cross-pollinate good ideas/successes and solution common challenges.

Agile Fundamentals

Agile is a "mindset" and a true paradigm shift in the way work is structured, planned, executed, and monitored. Agile requires significant and complex changes throughout the organization effecting leadership, culture, resources, and customers/end-users.

Traditional waterfall projects often rely on a detailed set of fixed requirements to execute a linear, multi-year development cycle, culminating in delivery of a complete solution at the end of the project.

Traditional development assumes that requirements are fully understood and documented at the beginning of a program and will undergo little to no change throughout the

development lifecycle. In practice, requirements change over time as organizational and program objectives change in response to constantly evolving threats and needs.

Agile begins with an understanding of the Agile mindset, which is driven by four values and twelve principles.

Leaders and teams must understand and fully embrace the mindset, values, and principles to realize the benefits of Agility. To truly be Agile requires significant changes to organizational culture, leadership approaches, and team dynamics.

How the organization initiates, plans, and executes work will also change and require leveraging new frameworks and practices that align to the Agile mindset, values, and principles.

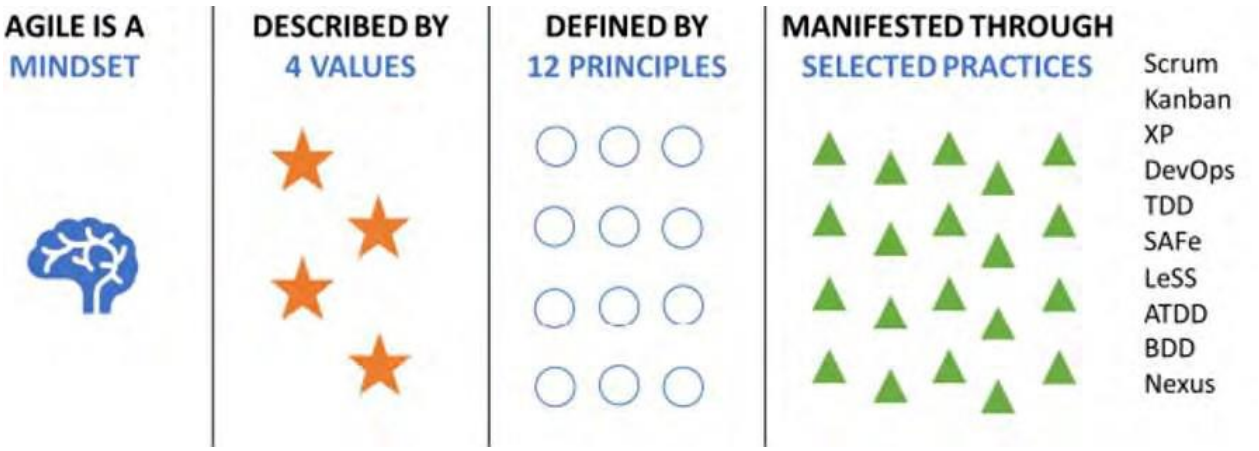


Figure 4: Agile Mindset, Value, Principles, and Practices

The Agile mindset emerged in 2001 after 17 industry leaders created the Agile Manifesto with the intent to design and share better ways of developing software. The foundation of Agile is a culture of small, dynamic, empowered teams actively collaborating with stakeholders (e.g., customers, end-users) throughout the development cycle.

Although we discuss Agile in terms of software development, many of the core principles emerged from lean manufacturing concepts and principles applied at companies like Toyota as far back as the 1930s. Therefore, Agile is highly applicable to the evolution of products and services outside of software development.

The Agile mindset is comprised of 4 Agile values:

Figure 5 – The Agile Manifesto

The Agile value statements above do not fully discount the items on the right but recognize the items on the left generate greater value and faster delivery of value to the customer and organization.

Agile values (items on the left) promote consistent delivery and demonstration of working software by an empowered teams that actively engages customers/end-users to enhance future value. Agile accepts the evolving nature of large, complex systems and acknowledges that upfront, predictive planning poses significant challenges to maximizing value delivery.

The incremental and iterative delivery of working product in Agile allows the team to actively collaborate with customers/end-users and capture fast feedback based on demonstrated of working product. This feedback generates new and shared knowledge that guides future direction and investment to maximize future value. Further, delivery in smaller increments results in smaller, less risky investments.

This stands in contrast to predictive (also referred to as traditional or waterfall) project management which places a heavy emphasis on the items right. Predictive project management assumes the organization has enough understanding on Day 0 to determine all requirements necessary to deliver at the end of a long-term (multi-year) horizon.

The organization leverages the Day 0 requirements to estimate schedule and budget, as well as contract vendors. This effectively locks in the Day 0 requirement and minimizes opportunities to adapt requirements based on new learning, stakeholder feedback, and changes in competitor capabilities.

Success becomes delivering the Day 0 requirements on schedule and at budget, with little to no consideration of what requirements generate value to customers/end-users.

Further, the completed requirements are delivered in bulk at the end of a long- term, sequential waterfall software development lifecycle (SDLC) so customers/end-users only get to provide feedback once at the beginning of the project (Day 0) and again when all requirements are delivered for user acceptance testing (UAT) at the end of the project.

This minimizes any opportunities to adapt requirements to enhance the value of the product delivered. If feedback is received at UAT that changes the original requirements, it may be challenging to implement due to cost, schedule, and contractual commitments.

Note: changes to requirements at this late stage may result in either lengthy contract renegotiation with customers/vendors or friction between the government and customers/vendors.

By contrast, Agile efforts focus on incremental delivery of work (value) based on a project vision and high-level objectives stated in terms of desired capability and outcomes.

This enables Agile projects to be flexible by letting the design emerge from the continuous learning that takes place at an accelerated pace as a result of continuous delivery of working capability based on user needs and emergent design principles as a project unfolds.

Agile embraces and harnesses change to maximize value to customers/end-users.

Agile anticipates new learning will occur, customer/end-user needs will change, and competitor capabilities will evolve.

It's approach to initiating, planning, executing, and delivering value is iterative, incremental, and highly dynamic/adaptable.

This avoids requirement lock-in until the last possible moment and only locks in a small set of requirements for during each short increment, allowing future requirements to adapt as needed to maximize value.

This requires a higher degree of user engagement and collaboration.

The Agile Manifesto identifies 12 Agile principles detailed below.

Note: Since Agile is leveraged across a variety of industries, products, and services outside of software development, you can add the words "product" or "service" wherever you see "software".

Figure 6 – The 12 Agile Principles

As shown in Figure 4, a traditional waterfall development approach takes place in linear phases based on a requirements-design-build-test-deploy cycle.

User feedback is generally captured once at the beginning and additional insight is not received until the full solution is developed and ready for testing.

In traditional waterfall the end-user receives no value for years.

In contrast, Agile development lifecycle promotes continuous delivery of small increments of working product, which provides opportunities for demonstration to users and capture of near real-time feedback within each delivery cycle.

This feedback should be used to guide future work effort to enhance/maximize future value. This iterative cycle realizes incremental value over shorter timeframes than traditional waterfall approach

(see Figure 5).

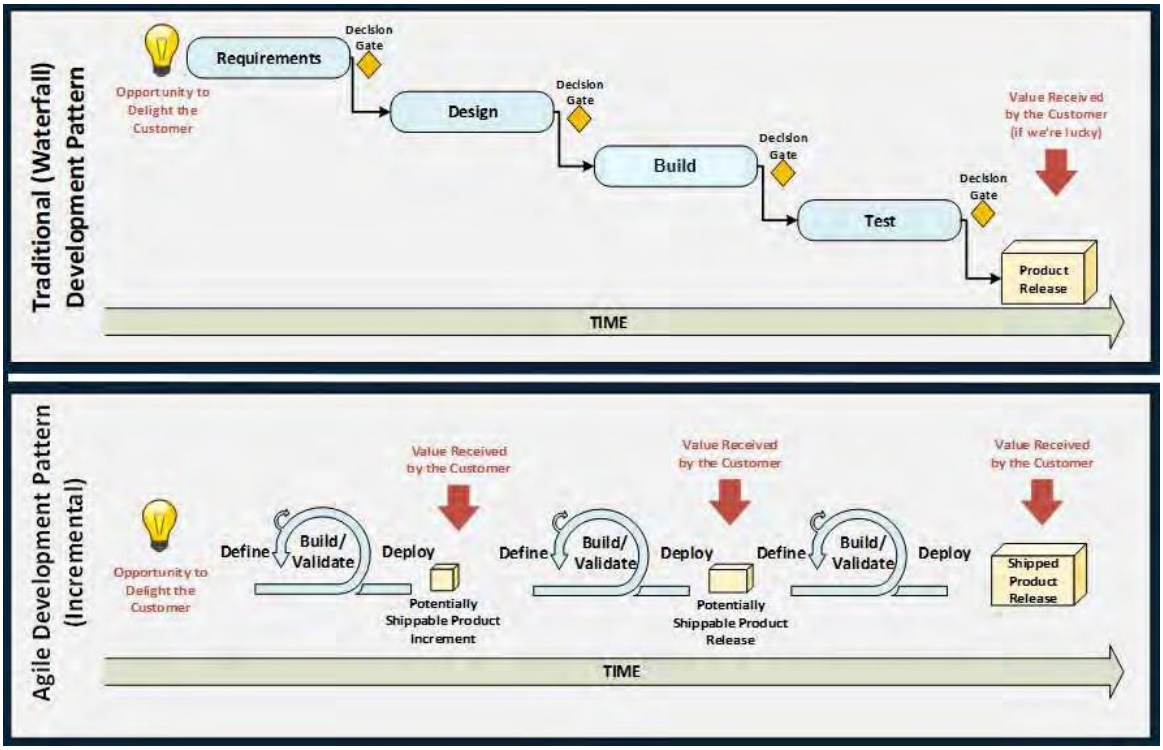


Figure 7 – Agile vs Waterfall Development Approach

The above illustrates a faster, more consistent delivery of value through Agility. By delivering work in smaller, more consistent increments, fast-feedback loops can be established to guide future direction and enhance/maximize value.

This stands in stark contrast to Traditional/Waterfall, where the big-bang delivery at the end of the project is based on Day 0 requirements and minimal customer feedback until all development activities are complete.

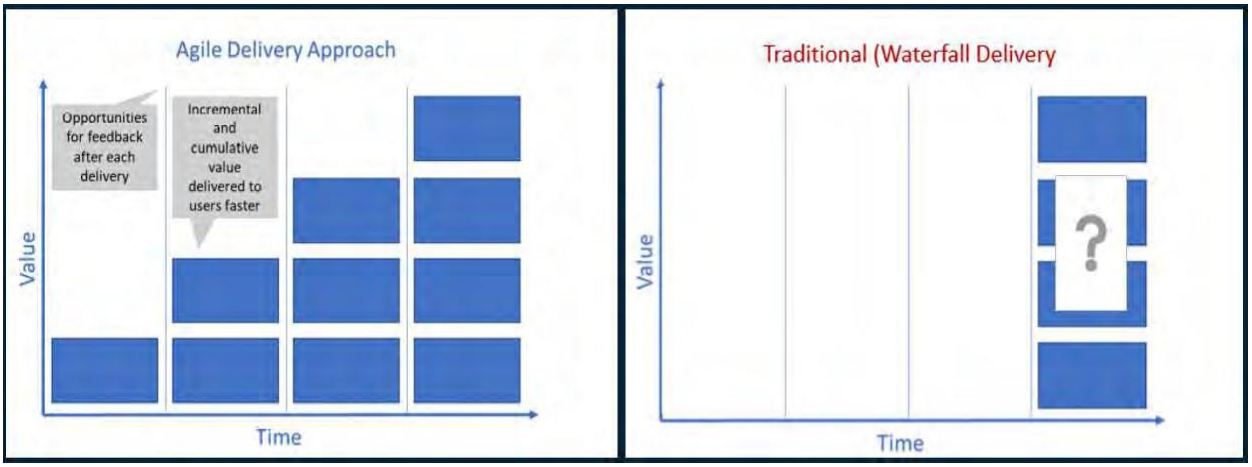


Figure 8 – Agile vs Waterfall Value Delivery

The table below highlights attributes of predictive/waterfall software development, the resulting challenges, and how Agile provides a solution for those challenges.

Important Note: In an Agile structure, optimally, the Government is accountable for the vision, roadmap, and prioritization of Agile backlog requirements, and must work with the contractor to ensure that the desired value is delivered.

Program Element	Predictive (Waterfall) Attribute	Resulting Challenge	Agile Solution and Benefits
Mindset	**Success = Adherence to the plan.** Emphasis on delivering all Day 0 requirements on time and at budget.	*The product lifecycle continues to shrink and teams need to regularly adapt to new learning, customer needs, and competitor capabilities.*	**Success = Maximizing value delivered and improve team performance.** Feedback routinely captured to adapt scope to maximize value.
Scope / Requirements	**Scope/requirements defined for multiple years on Day 0, when very little is known.** They are then used to estimate cost and schedule, develop CDDs, and contract vendors (which locks in requirements).	*Fixing scope early when the least knowledge is available about the product reduces opportunities to maximize value and increases the threat of building obsolete/ low-value requirements.*	**Scope and requirements constantly evolve to maximize value.** Product Owner routinely prioritizes, adds, deletes, requirements based on emerging customer/end-user needs, new learning, and evolving competitor capabilities.
Cost	**Exhaustive upfront analysis for initial investment because it is risky and spans multiple years.** Future investment decisions hinge on delivering at/under baselined budget.	*Attempting this on Day 0 when the least information is known greatly increases risk, does not allow us to incorporate new learning, and focuses us on managing to cost over value.*	**Smaller, incremental investment are less risky and value delivery data guides future investments.** Future investment driven by assessment of product value and team performance.
Schedule	**Exhaustive upfront long-term planning detailing sequenced tasks/activities required to deliver each and every requirement.** Durations for tasks/activities estimated. A multi-year integrated master schedule (IMS).	*Attempting this on Day 0 when the least information is known greatly increases risk, does not allow us to incorporate new learning, and focuses us on managing to schedule over value.*	**Long-term roadmaps set value-based targets, to allow team to focus on detailed near-term increment planning.** Planning is value-driven - defining "what", trusting the team to determine "how".
Defining and delivering value	**Value is assessed once at project initiation and rarely reassessed.** Initial assessment of value determines scope that is delivered big-bang multiple years into the future.	*Challenge: Business value changes over time but Teams build based on fixed Day 0 requirements. This generates significant risk customers won't value what is built.*	**Product Owner routinely assesses value and prioritizes work accordingly.** Routine user engagement, feedback, and value assessments guide future effort.
Leadership Focus	**Top-Down Management**: **Managing teams** to deliver defined scope at set schedule and budget developed with the highly detailed tasks provided by the team explaining each step they will perform.	*Challenge: Does not empower and trust the team. Ignores emerging customer needs, new learning by the team, and evolving competitor capabilities.*	**Servant Leadership**: **Empowering teams** to figure out what is valuable and how to best deliver it.
Team Focus	**Adhere to the long-term plan and satisfy phase gate approvals.** Effort invested in product documentation to satisfy phase-gates and	*Challenge: Plan adherence may not equate to delivering value. Phase gate approvals require significant LOE that distract the team from delivering valuable product.*	**Adapt requirements to maximize value delivered and improve value delivery.** Effort focused on delivering valuable product and

	progress to long-term, big-bang delivery.		minimizing administrative work that distracts.
Release (value delivery)	**Big-bang, multi-year delivery after a black box build.**	*Challenge: Sponsors/ customers make big upfront investments and wait a long time for value delivery.*	**Value demonstrated and delivered incrementally in the shortest time possible** which generates feedback to guide future effort.
Customer feedback / collaboration	**Heavy focus at the beginning (requirements capture) and at the end (user acceptance testing)** generates risk of significant rework and/or unsatisfied customers.	*Challenge: Long delivery times and minimal customer engagement throughout results in a disconnect between what the customer initially requested and what they need now.*	**Regular customer/ user engagement and collaboration to maximize value.** Demonstration of working product at the end of each increment generates fast feedback to guide future direction.
Risk / Uncertainty	**Locking in requirements over the long-term on Day 0 increases uncertainty, complexity, and risk.** Teams may fail to meet customer needs and remain unaware until the end of the project.	*Challenge: Because delivery is over the long-term, significant uncertainty, risk, and complexity is generated, resulting in the need for large contingency/ management reserves and a greater risk of failure.*	**Incremental delivery in smaller batches reduces uncertainty, complexity, and risk.** Shorter timeframe results in less risk, more information about each risk, and less required contingency funding.
Team Structure	**SMEs with specific knowledge/skills that are not highly interchangeable.** Team members borrowed from functional areas.	*Challenge: Single points of failure and capacity challenges for specific resources can severely impact progress.*	**Teams with plug-and-play skillsets** that are interchangeable. Dedicate core team members to help minimize distraction.
Testing and Evaluation	**Extensive effort put into developing test planning documents, manually performing multiple rounds of end-to-end testing.** Test and evaluation engaged once all of development/build activities are completed.	*Multi-year build effort and manual testing increases LOE/time required, complexity and risk. Multiple testing cycles increase LOE substantially since each requires full manual testing effort.*	**Incremental releases and automated testing reduce testing LOE, complexity, and risk.** Developers build automated tests as part of normal build cycle and definition of done.
Quality	**Quality assessed when big-bang delivery occurs.** Disconnect between build activities and when quality is assured. Multi-tasking, deadline driven development, and manual testing are norms.	*Sequential nature of SDLC and necessity that all requirements are completed before testing and evaluation may result in substantial rework. Multi-tasking, pressure to meet deadlines, and manual testing further erode quality.*	**Quality assessed routinely as smaller increments are delivered.** Incorporating fast feedback, automated testing, continuous improvement, work in progress (WIP) limits, result in higher quality work.

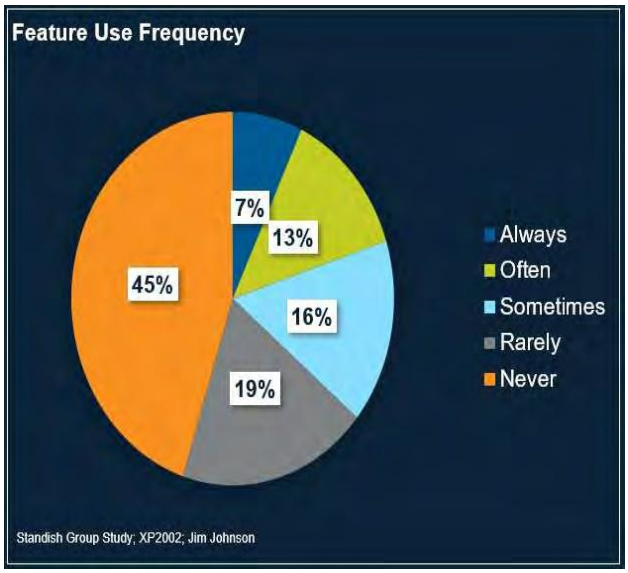
Table 2 – Agile Solutions to Inherent Waterfall Challenges

Connection to Lean

Agile and DevSecOps practices are based on Lean manufacturing principles, which is why they are frequently applied to the development of products and services outside of IT/software development. The goal of lean is to relentlessly minimize waste through the application of lean principles:

Figure 9 – Lean Principles

- **Deliver (value) quickly:** Manage the flow of work to increase the pace of feedback and opportunities for learning. To be able to deliver quickly, the organization must be aligned on leadership support, training and staff capability, and a clear understanding of how to apply Agile and DevSecOps practices.
- **Eliminate waste:** Minimized non-value-added work effort (e.g., administrative tasks, documentation, phase gate preparation activities), reducing "gold plating" (overbuilding based on assumptions of desired user value); delivering in small batches to reduce complexity (and the possible rework resulting the complexity associated with large batches).
- In Agile, flexible requirements and regular, fast feedback allow the team to steer effort towards high value needs of customers/end-user and away from no- or low-value work. This is different from traditional waterfall projects where requirements are locked in on Day 0 when there is limited knowledge. The inflexible nature of waterfall and minimal customer/end-user engagement results in limited understanding of customers emerging needs/requirements. Further, the inability to adapt to new learning can result in significant LOE focused on no- or low-value requirements. The Standish Group study noted that approximately 64% of features built were rarely or never used (building the obsolete). Note: Adopting a demonstratable MVP to iterate from provides a mechanism to gain valuable customer/end-user feedback that can help eliminate waste.



- **Build Quality In:** build quality at the source of the work to ensure quality is produced every time. This means that all known quality requirements (including T&E, security, and operational) are transparent to the developer and that all quality processes (including T&E, security, and operational) are pulled to the left to align with Agile build cycles. Standardizing and automating as much as possible to avoid issues resulting from inconsistent application of testing and quality measures will help reduce human error (e.g., environments will remain consistent if applied consistently). Further, allow the process to be stopped to determine why quality defects occurred to avoid the same issue in the future.
- **Create Knowledge:** establish a culture of learning based on the data acquired during short delivery increments results in the ability to experiment and fail fast. The new learning from experimentation results in knowledge that can be shared across the organization. A safe and blameless culture helps to ensure that teams are encouraged to experiment and share both successes and failures for the benefit of the organization as a whole and/or other teams. People and teams learn as much (if not more) from their failures, so dissemination of failures can help other teams avoid those challenges in the future.
- **Defer Commitment:** maintain the ability to make decisions until the last possible moment (when the most information is available to inform the decision). In traditional waterfall, most decisions are made on Day 0 when little information is known to inform decision-making. The flexibility and focus on near-term planning afforded by Agility results in the ability to make key decisions in the future when the most information is available to inform decision-making (including in areas such as requirements, technology, design, and approach).
- **Deliver Quickly:** Work only has value once working (usable) product or service is delivered to the customer. Work delivered to the customer generates value for the customer and also value for the team in the form of customer/end-user feedback (assuming feedback loops have been established). In traditional waterfall, value is delayed to the customer (often for multiple years) and feedback to enhance value is minimal/non-actionable (captured at the tail end of a multi-year development effort during user acceptance testing).
- **Respect People:** Creating a culture that is focused on valuing and respecting people. This includes respecting trusting/empowering teams to get the job done (respecting their subject matter expertise and work ethic), pushing down decision-making to the lowest level (where the best information/knowledge exists), establishing a culture of continuous learning, creating a safe and blameless culture where the focus is on solutioning over blaming (to allow teams and individuals to experiment, take calculated risks, and learn/share knowledge openly).
- **Optimize the Whole:** Take a data-driven and scientific approach to optimizing product/service delivery (output efficiency and quality). This requires end-to-end understanding of the value stream (people, systems, and processes) required to deliver to your customers and end-users. It also requires leadership and teams to understand both the current state and desired future state. This helps to identify and ruthlessly attack non-value-added activities and blockers/bottlenecks to delivering value. Optimizing the whole requires making data driven decisions based on testing specific hypothesis and iterating based on the learnings.

Connection to DevSecOps

While Agile addresses the organization of work around value delivery, enhance development team performance, and shift to an adaptable customer-centric approach to maximize value, DevSecOps enhances the efficiency of the architecture path (pipeline) to allow for the delivery of working software at the speed of Agility and pace of the development team.

DevSecOps promotes continuous integration, continuous delivery of work by encouraging the team to look at the end-to-end flow of value delivery, or the value stream (see Figure below). This end-to-end flow begins with the initial request and progresses to delivery to operations/customers and beyond. This requires that development, testing, security, and operations teams work together at the speed of value delivery. This requires testing, security, and operations teams to:

- Work with development teams early and as often as possible.
- Provide their needs and requirements upfront and with complete transparency
- Shift involvement to the left - at end of each increment instead of at the end of all development.
- Adapt their processes/approaches to move at the speed and cadence of the delivery team.
- Provide regular input to the Product Backlog, including Stories and/or acceptance criteria.

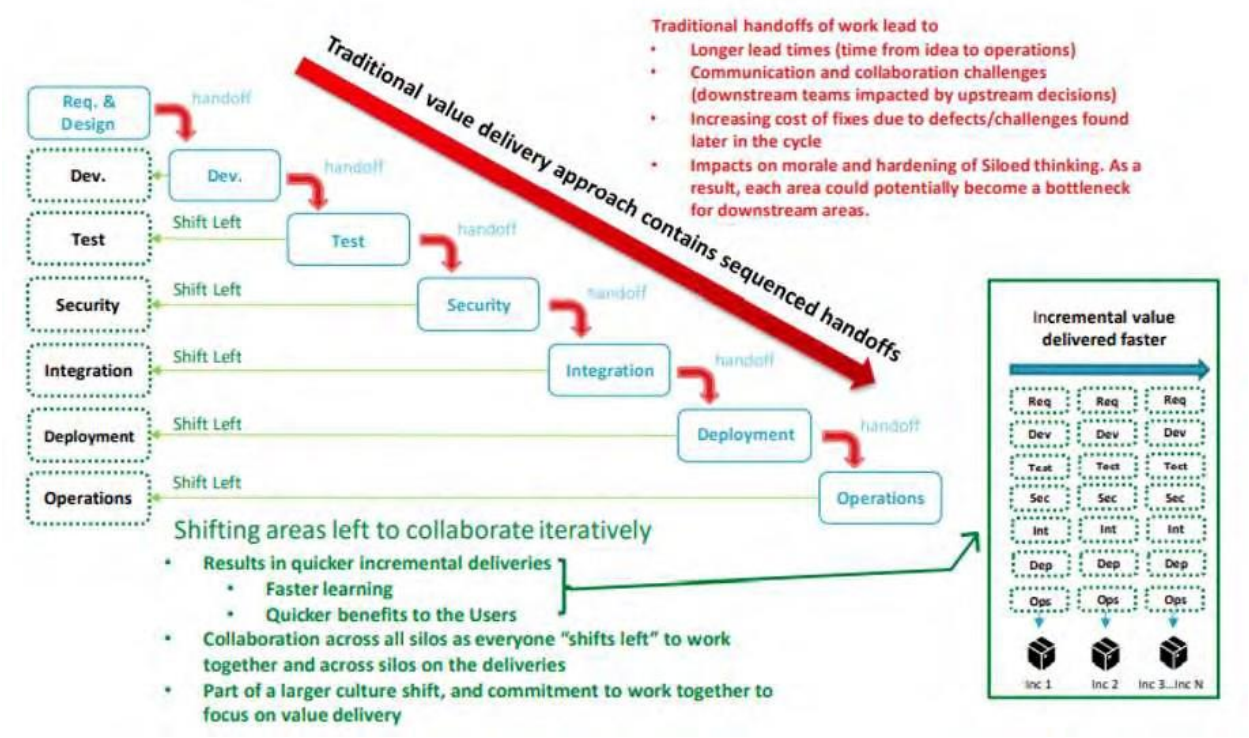


Figure 11 – DevSecOps Shifts Development Activities Left

DevSecOps encourages the use of cloud technologies to automate all aspects of development, testing, integration, and deployment. This makes end-to-end delivery faster, repeatable, and less error prone. Automation may be used for:

- Build/coding activities
- Testing (developing an automated test for a Story as part of the definition of done)

- Continuous Integration
- Deployment
- Environment instantiation
- Security monitoring and remediation

DevSecOps provides many benefits, including:
- Targeted automation can result in lower LOE investment, higher quality results
- Shorter times from idea to delivery to customer and operations (shorter lead times)
- Faster feedback and learning enhances value, minimizes waste, identifies defects early and reduces costs to remediate defects
- Less siloed work functions yields higher team morale (everyone is in this together!)
- Collaborative problem solving that benefits everyone/optimizes the whole
- Shifts solutioning and decision-making away from what is best for one silo/function to what is best for the organization as a whole
- Enhance communications reduce surprises due to upstream/downstream impacts

Key challenges to implementing DevSecOps, including:
- Organization and functional resistance (desire for change)
- Ability to effectively manage change (people, process, systems/tools)
- Workforce development and ongoing coaching
- Ability to commit to continuous improvement

Structuring Work Around Products and Value

Agile at the organization level emphasizes aligning resources to delivery and continuous evolution of products that are valued by the customer/end-user. The goal is to determine what products the organization's customers most value, which should inform investment decisions. Then the value stream (steps to produce) for each valuable product must be determined. The value stream should encompass all steps required to deliver the product to the customer. Product generation may require contribution(s) from different functions or silos within the organization and the value stream map should reflect those handoffs. Value stream mapping of all products creates a visualization of product delivery in the form of an overlayed matrix spanning functions within the organization. Understanding the value stream allows the organization to relentlessly identify and eliminate bottlenecks/impediments to delivering value to the customer/end-user. Note: Value- and customer-centric organizations may choose to break down functional silos and reorganize people, processes, systems around products. This may result in repurposes resources away from no- or low-value activities that do not create value for the customer/end-user.

Agile at the team level structures work to generate stand-alone value (not tasks/activities). The goal is to produce Stories that have stand-alone value and are delivered (complete design-build-test activities) within each sprint/iteration (delivered in 2-4 weeks). This is a distinct from waterfall, where work is structured in a series of tasks/activities required to complete the work and delivered once development is complete for all requirements (delivered in multiple years). Agile work structuring provides the ability to change and adapt to emerging customer needs,

new learning, and evolving competitor capabilities. It

also helps ensure that value is delivered early and often. If Agile teams do not structure work around value or are pressed by leadership to provide tasks/activities, they risk falling back into waterfall practices that will inhibit the value and benefits of Agile (see below for an explanation why).

Traditional (waterfall) work structure: traditional waterfall projects attempt to capture, fix, and time- box all work (in the form of requirements) over multiple years. Each requirement is broken down into all necessary steps (tasks/activities) and these steps are sequenced along the SDLC phases (plan, design, build, test, deliver, etc.). The task/activity information is used to estimate schedule and budget, while the requirements are used to drive contracting. All of this information informs the creation of an Integrated Master Schedule (IMS) used to manage schedule, budget, and delivery of requirements.

This approach requires that teams make risky assumption(s) on Day 0 (when little is known):

- They understand all requirements necessary to satisfy the customer
- That requirements won't change much (even over multiple years)
- That they understand each requirement well enough to break them down into sequenced tasks/activities required to complete the work
- That they understand each requirement and the related tasks/activities well enough to provide accurate estimates of time and cost

Developing and baselining the IMS locks in requirements, scope, schedule, and budget. It makes the general assumption that requirements are accurate, and scope will be delivered and with minimal cost/schedule variance. The IMS does not measure value delivered, just progress towards delivery of the Day 0 scope (which may or may not be valuable years into the future, since value changes over time). If variance occurs (sometimes as little as 10% variance), the team may be required to engage in a time- consuming and high LOE replanning effort (that distracts from delivering value). Further, the IMS shifts leadership focus away from maximizing value and to plan adherence along pre-defined scope, cost, and schedule. Changes to baselined requirements and IMS (even to minimize waste/maximize value) is perceived as negative/a failure on the part of the team. If the requirements and related IMS are established via a contract with a vendor, the impact of changes is extensive contract negotiation and the vendor shifting cost and schedule to the right to accommodate change.

Agile's value-centric work structure: In contrast, Agile planning efforts are value-centric (not task/ activity-centric). Agile planning is focused on determining "what" is valuable and then prioritizing that work in a manner to maximize value. Since value delivered is a primary measure of success, all work (long-, mid-, and near-term) is structured to deliver standalone value. "How" things will get done (the tasks/activities) are left to the discretion of trusted/empowered team members. Therefore, standard and best-practice Agile breaks work down to the Story level – a piece of work that delivers value. It does not break work down below that level to detail the task/activities/steps required to deliver that
value. Instead, leadership will gain significant insight into the value delivered by the team, the team's ability to deliver value in the future, and what the customer values/prioritizes right now. This change requires leadership to shift away from "managing/monitoring" the performance of tasks/activities at cost/schedule elements and towards adoption of "servant leadership" where

the focus is removing blockers/impediments to help the team deliver more value faster. Agile also pushes Teams to think differently about how work is structured – in Stories that deliver stand-alone value and encompass plan, design, build, test, deliver within a single Sprint/iteration. Note: Some warning signs that work is structured inappropriately: 1) too many stories fixated on planning; 2) stories express steps towards value (e.g., tasks/activities) instead of something valuable that can be delivered; and/or 3) stories attempt to recreate the waterfall SDLC.

Agile Planning

Long-Term Agile Planning: The Agile product vision is expressed in a Capability Needs Statement, SW-ICD, or similar document, which shares an understanding of where we are now (mission), where we want to be in the future (vision), the capabilities and value delivery strategy required to get the product from mission to vision. The capabilities should be captured in a manner that answers the question of "why" the capability is valuable, expresses both the current and the future state of each capability, and so it has stand-alone value (is not dependent on other capabilities). All capabilities should be prioritized in terms of anticipated value delivery to customers/end-users (approaches like weighted shortest job first can be useful to guide prioritization).

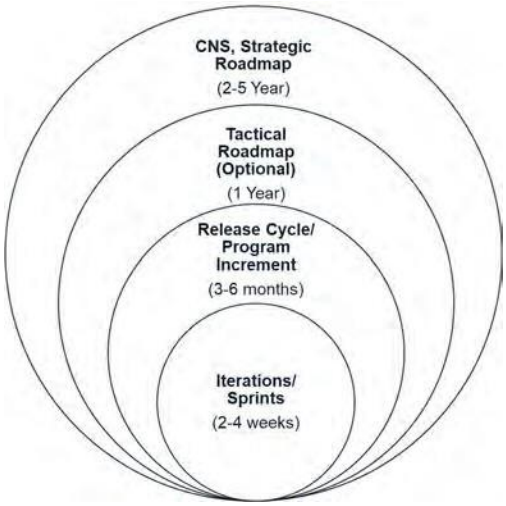


Figure 12 – Agile Planning

The Agile Strategic Roadmap is a visual tool that shows the targeted delivery of value over time designed to progress the product from mission to vision. The roadmap is intended to change and adapt over time to reflect feedback received by customers/end-users, new learning, and competitor capabilities.

Because Agile roadmaps are defined at a higher level of abstraction, teams have the flexibility to delay decision-making until the team is ready to develop a specific capability or work on a specific Epic/Feature of that capability. This flexibility is built into the Agile process to enhance and maximize value. Further, since change is expected over the long-term, generating detailed planning that speak to "how" each capability will be delivered in full (detailed requirements, tasks/activities) is consider wasteful planning. Capturing detailed tasks/activities is also considered lack of trust that disempowers the team. Note: The initial roadmap should identify the capabilities/features required to get to MVP.

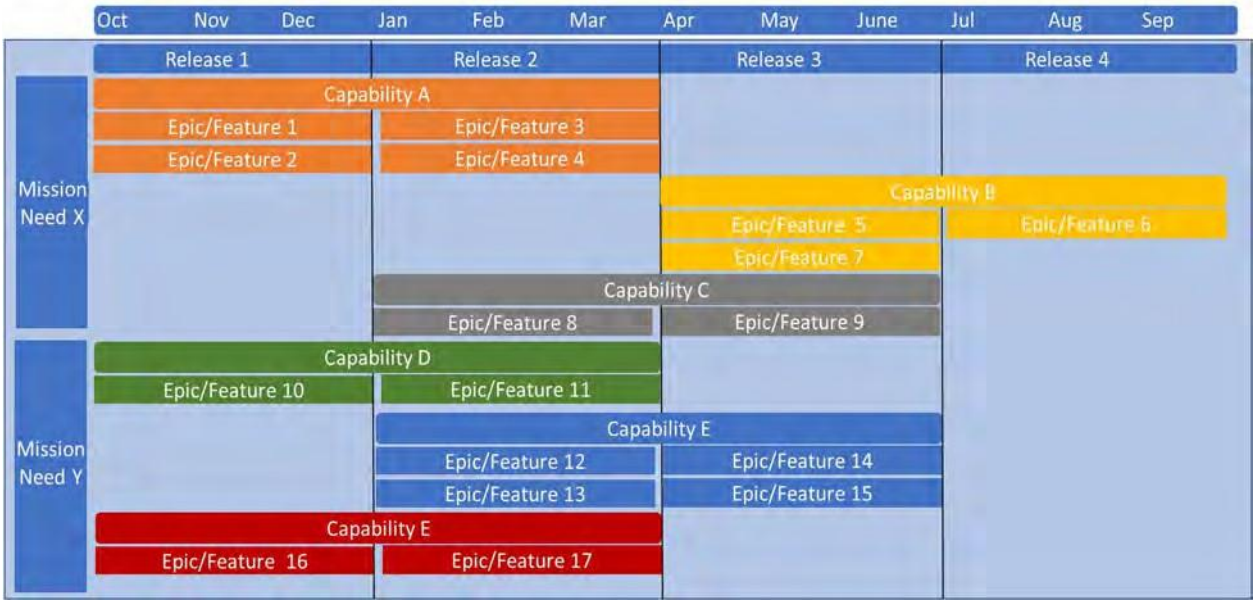


Figure 13 – Agile Roadmap

Mid-Term Agile Planning: A tactical roadmap may be useful to identify capabilities or work that demonstrates progression towards delivery of a capability along a mid-term view (i.e., a timeframe aligned to annual budgeting and/or reporting cycles). Capabilities can be broken down into Epics/Features targeted for delivery within this mid-term timeframe to provide greater fidelity but are kept at a higher level of fidelity with an emphasis on anticipated value. Epics/Features should be structured to have stand-alone value and move us towards delivery of a capability. Epics/Features should be prioritized in terms of anticipated value to customers/end-users. Greater emphasis may be placed on Epics/Features targeted for delivery over the next release cycle/program increment. Note: Epic/Feature in this roadmap are targets that may change and since change is anticipated, detailed planning or discussion of how all of the work will be completed (tasks/activities) is considered wasteful planning.

Near-Term Agile Planning: Planning the above at a higher level allows the Team to focus on the near- term (the upcoming program increment and/or release cycle which is typically 3-6 months). In the near- term, the team focuses on breaking down the targeted Epics/Features into all known Stories and committing to delivery at the end of the program increment/release. The Stories generated express what the customer wants and why each Story is valuable but does not express how the Story will be delivered (tasks/activities). In Agile, each Team member is trusted and empowered to complete work using whatever "Definition of Done" is agreed to by the Team.

Detailed Tasks and Activities Are No Longer Required: As you can see above, work is structured differently in Agile and detailed tasks/activities are not generated at any step in the planning process. While individual team members may create their own checklists of tasks/activities for Story completion, they are not required to do because they are trusted and empowered. Further, capturing this information skews/disrupts Agile automated metrics and reporting typically found in best of breed Agile tools and used for continuous improvement purposes. Therefore, Agile teams cannot generate detailed schedules without creating two competing program management frameworks that increase overhead

for the team and distract the team from delivering value. Instead, Agile leaders focus to Agile metrics to guide improvements in value delivery and team performance.

Minimum Viable Product (MVP) and Minimum Viable Capability Release (MVCR): Variability and risk are inherent in technology programs. Because of this, developing and measuring an MVP provides an opportunity to evaluate the solution, obtain results, and reduce risk before committing to the investment of the entire system. MVP focuses on developing the simplest (lowest LOE) form of the product (the minimum) that customers/end-users find valuable (viable). **Demonstration of working product (starting with the MVP) provides opportunities for customers/end-users kick the tires and share valuable feedback to provide insight on future direction and effort, including feedback if the product/service isn't needed, negating the need for future iterations, and wasted investment.**

- **MVP:** An early version of the software to deliver or field basic capabilities to users to evaluate and provide feedback on. Insights from MVPs help shape scope, requirements, and design.

- **MVCR:** The initial set of features suitable to be fielded to an operational environment that provides value to the warfighter or end user in a rapid timeline. The MVCR delivers initial warfighting capabilities to enhance some mission outcomes. The MVCR is analogous to a minimum marketable product in commercial industry. It should be as small as possible and capable of rapid validation.

The key to producing MVP is to find a balance between minimum and viable – a product requiring minimum effort on the part of the team and the creation of a viable product (that customers/end-users can actually use). MVP begins the process of continuous value delivery and iteration that reduces the risk inherent in large, long-term delivery. Future iterations (post MVP) evolve the MVP by adding functionality/capabilities valued by customers/end users that help mature the product.

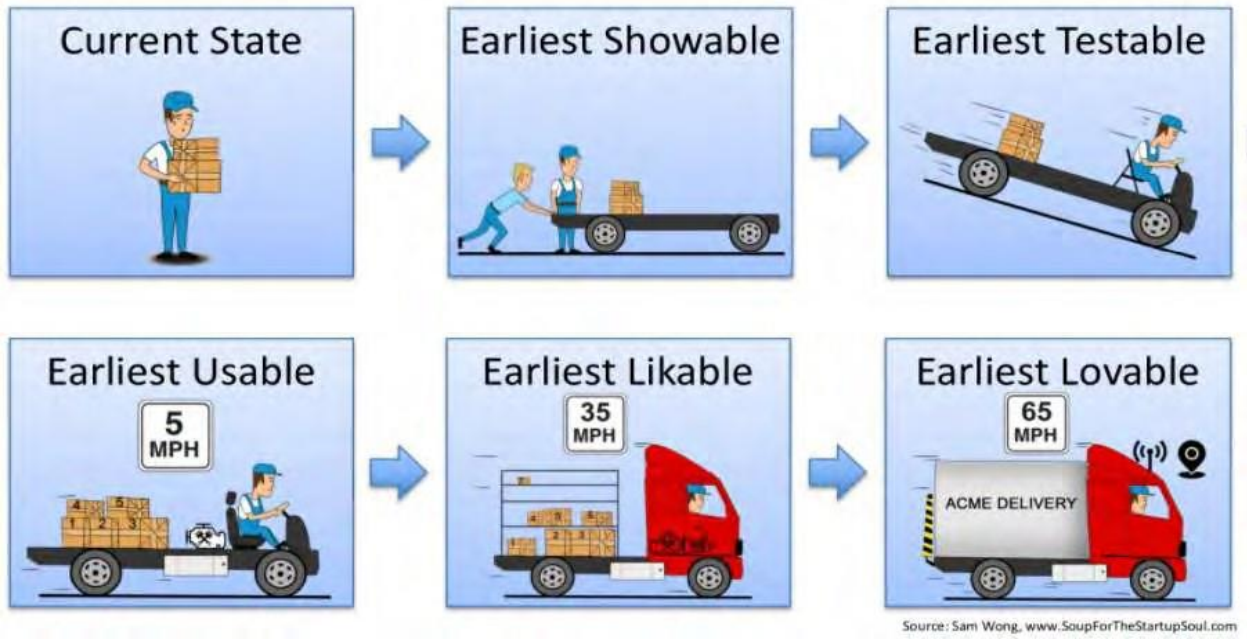


Figure 13 – Minimum Viable Product

The MVP provides a baseline set of capabilities to test assumptions of the proposed system and gather appropriate data to determine if the proposed system is delivering the expected or acceptable end-user value. Common guidance in industry indicates the MVP should be sized as a manageable and demonstrable set of capabilities. Feedback received from end-users is critical to guide future iterations.

Common Roles/Responsibilities

Agile roles and responsibilities require a shift in thinking. Waterfall investment is made in long-term projects performed by resources borrowed from and shared with stove-piped/siloed functions. In Agile, this shifts to shorter, incremental investment in high value products and high performing teams. This requires identification and focus on what products actually generate value to customers/end-users and establishing dedicated, high-performing teams to continuously evolve those products in ways that maximize value.

Critical to ongoing product evolution is establishing feedback loops with customers/end-users to better understand what they value. Since what the customer/end-users may change over time, frequent reprioritization of requirements (Stories) is required to be Agile. This is a distinct departure from traditional waterfall development where customers/end-user are engaged once at the beginning of the process (initial feedback via requirements gathering) and once at the end of the process (feedback via user testing).

Monitoring and adapting based on new learning via this continuous customer/end-user feedback loop requires a role established to be the "Voice of the Customer" – the Product Owner. The Product owner should have an understanding of the various customer/end-user communities and the authority to make decisions about future vision and direction of the product, including the priority of features/stories for the team. The Product Owner works closely with the Scrum Master and Team to ensure progress and the delivery of high value work, accepting and rejecting delivered stories and determining when to release to customers/end-users. The Product Owner is frequently a government employee from the user organization that owns the requirements. While some programs have a contractor fill this role to work with Government users, it is recommended that this role be filled by a government person that maintains accountability over ensuring that the proper business value is delivered. This will require a greater commitment on the part of the Government to actively participate in the day-to-day Product Owner-owned activities.

The Scrum Master supports the Product Owner and the Team as a "servant leader" who may coach/mentor both the Product Owner and Team on sound Agile practice. The Scrum Master leads all Agile ceremonies and champions the process to ensure value is delivered efficiently and effectively to the customer. The Scrum Master acts as a "servant leader" to help the team remove bottlenecks/blockers, which may include appropriate escalation. The Scrum Master may be a government role or may be facilitated by a member of a contractor team.

The Core Team consists of empowered subject matter experts that can plug and play on the work to be performed. Each team member estimates work effort on a relative basis, helps determine how much work the team can complete each Sprint, and delivers (design-build-test) high quality Stories. Team members also support continuous improvement efforts in retrospectives. Note: SWP and DoD application roles may differ slightly due to unique operating environments and specific needs of the program.

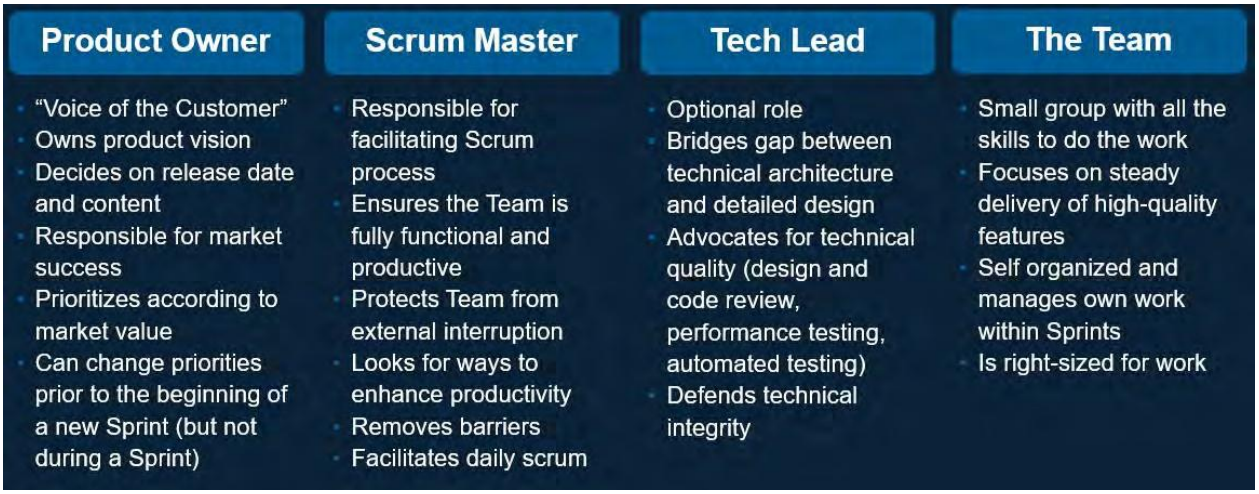


Figure 14 – Common Agile Roles

The core team above is supported by an extended team that will adapt based on the operating environments and specific needs of the program. The extended team may include primary stakeholders and representatives of functional support activities, to include the acquisition leadership, contracting, test, certification, and accreditation (C&A), the user community, external systems, and organizations contributing financial support. These team members are organization and program specific. Early engagement with this broader team to help them understand the Agile process and how they can integrate/adapt their existing people/processes to meet Agile program needs is critical.

Common Tools

A few tools and techniques used to track Agile work and requirements have gained in popularity from their alignment with Lean-Agile principles:

- **Roadmap** – A highly flexible and adaptable visualization of strategic objectives/themes that may be achieved over the long-term (multiple years). The goal is showing progress from mission (where we are today) to vision (where we want/need to be in the future). Items on the roadmap defined in terms of value but are not planned at a significant level of detail (detailed planning is focused on the near-term). Roadmaps replace detailed, multi-year requirements documents and long-term project schedules/ integrated master schedules (IMS) consisting of detailed tasks/activities.

- **Backlogs** – A dynamic list of prioritized user stories. Backlogs can be managed for the overall program, each product, each release, and each sprint. A backlog is a near-term view of work that delivers stand-alone value (can be released/delivered to the customer/end-user. Agile requirements are broken down into Features/Epics that deliver stand-alone value, then into Stories that build toward the completion of Features/Epics. Work is not broken down into tasks/activities. Instead, it expresses "what" the customer wants and avoids "how" to complete the work. The team is trusted to determine the "how". Types of backlogs may include the Product Backlog (all customer requirements broken down into stories), Increment or Release Backlog (what work the team is targeting to release), and Sprint Backlog (what stories the team intends to complete in the upcoming Sprint). Each type of Backlog contains the Agile requirements for the given product, increment, release, or sprint respectively. Backlogs replace detailed, multi-year requirements documents and long-term project schedules/integrated master schedules (IMS) consisting of detailed tasks/activities.
- **Boards**: Boards are frequently used to visualize and track ownership of work items and the flow of work. Boards show work prioritized by the Product Owner (e.g., backlog), work in progress, work that is blocked, and work completed. Visualization allows teams to quickly and easily determine ownership and the current status of each work item. Tools capture work related data and provide metrics that can be leveraged for continuous improvement. Boards may also show blocked work and work dependencies. Boards replace task assignments found in project schedules/integrated master schedules (IMS), word-of-mouth / email tasking, and tasking in project status meetings.

Measuring Progress and Success

Agile measures progress and success in terms of value delivered to the customer and the team's ability to deliver future value. Due to this, Agile metrics and usage are distinct from traditional waterfall metrics and usage (centered on cost/schedule). Agile metrics will require new ways of thinking and training for leadership and teams to maximize value.

- **Agile measures value delivered and ability to deliver value in the future:** In waterfall, success is measured based on adherence to the planned fixed scope and hitting schedule/budget estimates. Little time or effort is spent measuring the value of what is delivered to the customer/end-user or overall team efficiency. In Agile, scope is expected to change/adapt, and success is measured based on the amount of value delivered and team's ability to deliver value in the future. This shift in focus requires organizational leadership, project management offices (PMOs), and product team members to understand new metrics (velocity, burndown rates, cumulative flow) and jettison old ways of measuring progress. See metrics and related guidance, as well as value assessment guidance and templates.
- **Agile metrics are actionable, real-time, and low effort to produce:** In waterfall, metrics require significant investment of time and effort to produce (usually by the PM), which adds to administration and information lag. In Agile, metrics are generally automated and

real-time, because the underlying data is easily updated daily by team members. Access to this real-time data is provided via dashboards that anyone can access (increased transparency), making the data highly actionable.

- **Agile metrics generate transparency:** In waterfall, development occurs in a relative black box and the results of SDLC activities aren't available to customers/end-users to assess until the end of the project (during user acceptance testing). This approach minimizes transparency and creates challenges assessing progress, team performance, and quality/value of what was delivered. The highly manual reporting process provides regular opportunities to game results. Agile addresses these challenges by delivering value in shorter, regular cycles which allows frequent opportunities for measurement, experimentation, and calibration. These frequent deliveries provide near complete transparency and real-time insights that can be leveraged to assess team performance, ensure quality is built in, and measure value delivered. Automation of metrics results in minimal ability to game the system.
- **A blameless and safe culture allows for experimentation and knowledge sharing:** Agile emphasizes continuous improvement of performance and value delivery through frequent retrospectives. It also promotes a scientific approach where experiments are developed and applied to upcoming increment(s). The short duration of the increment allows the team to quickly determine if it succeeds (and becomes part of the team's DNA) or if it fails (the resulting failure occurs fast and has low costs associated). Agile recognizes that organizations and teams learn much from both successes and failures, so both should be shared far and wide to help other Agile teams improve value delivery. However, knowledge sharing can only occur in organizational cultures that are blameless, safe, and focused on solutioning. In cultures that are not blameless and safe, the fear of negative repercussions for failure may inhibit experimentation and knowledge sharing gained from failure.

Common Agile Frameworks

There are many Agile frameworks, each offering methodologies that apply variations of iterative development and continuous feedback. The most popular Agile frameworks include:

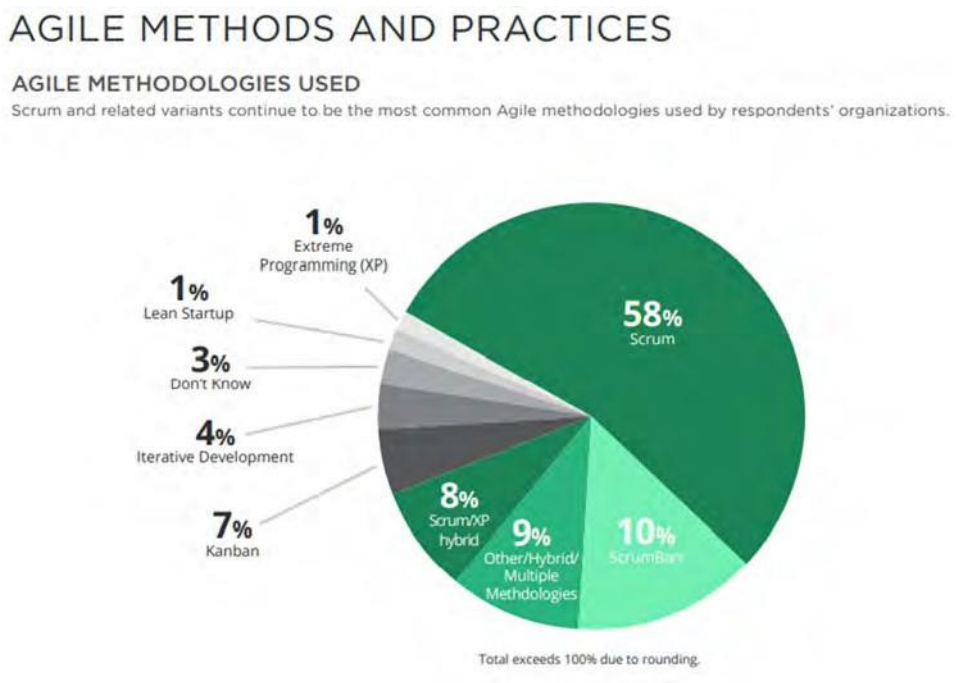


Figure 15 – Agile Methods and Practices

- **Scrum** – A lightweight, simple framework for teams to collaborate incrementally and iteratively on delivering value to the customer. See Section 2.10 for more information on Scrum.
- **Kanban** – A framework that enables visualization of the flow of work and allows the team to monitor work in queue, work in progress and the overall flow of work from inception through completion.
- **Scrum of Scrums** – A scaled version of Scrum, in which multiple Scrum teams work together on a large project or program.
- **Extreme Programming (XP)** – An Agile framework that focuses significantly on engineering and development practices to bring value to the customer by producing higher quality software. The five values of XP are communication, simplicity, feedback, courage, and respect.7 Practices such as pair programming have been popularized by XP.
- **Scaled Agile Framework (SAFe)** – A framework intended mostly for larger projects and programs that is based on Lean-Agile principles and addresses five core competencies: Lean-Agile Leadership; Team and Technical Agility; DevSecOps and Release on Demand; Large Solutions (Business Solutions and Lean Systems); Lean Portfolio Management.
- **Additional Agile frameworks:** Large-Scale Scrum (LeSS), Dynamic Systems Development Method (DSDM), Disciplined Agile Delivery (DAD), Nexus Framework by Scrum.org

Two very common frameworks are Scrum and Kanban. Take a moment to watch this video discussing each approach and their distinctions. Scrum is by far the most common approach. Below is an example and discussion of the ceremonies and roles/responsibilities related to Scrum.

How to Get Started – An Example Using Scrum

Scrum is by far the most popular form of Agile. Scrum emphasizes timeboxing value delivery in Sprints (typically 2-4 weeks in duration). When enough value builds up over multiple Sprints, the Product Owner schedules a Release of value to customers/end-users.

Sometimes when teams are getting started, there is value to establishing a standard release cadence (3-6 months) to identify bottlenecks/impediments in the release process.

Work for upcoming Sprints/Release is captured in the form of Stories that express "what" customers wants but not "how" the team will complete that work. Stories are contained in a Product Backlog that is routinely groomed (Stories are modified, added, deleted, reprioritized) by the Product Owner (the "voice of the customer") before a Sprint begins.

Once the Product Backlog is groomed, the team estimates the relative complexity of each Story and assigns Story Points. The team uses those Story point estimations to plan what they can complete in the upcoming Sprint.

The number of Story Points the team thinks they can complete sets the target velocity. This velocity may be based on the number of Story Points that the team completed in previous Sprints (actual velocity). The team pulls the targeted Stories into the Sprint Backlog and begins their Sprint.

During the Sprint, the Product Owner can continue to groom the Product Backlog items but may not add or change items in the Sprint Backlog. This protects the team from distraction. The team leverages daily standups/scrums to identify who owns each Story, the status of each Story on the board (for example In Backlog, In Progress, Blocked, or Done) and if anyone is blocked. The team does not use this time to provide status updates.

When the Sprint timebox ends, all Sprint Backlog items that are not completed move back into the Product Backlog. The team then demonstrates completed Stories to the Product Owner and possibly other customers/end-users to capture fast feedback to guide future effort.

The Product Owner may accept or reject Stories completed by the team. Rejected Stories will move back into the Product Backlog. After the demonstration, the team will reflect/retrospect on activities they want to Start-Stop-Continue.

This drives continuous improvement by highlighting what went well that they would like to continue, what didn't go well that they would like to stop/improve, and things they haven't tried yet that could improve performance (start). This framework is illustrated below.

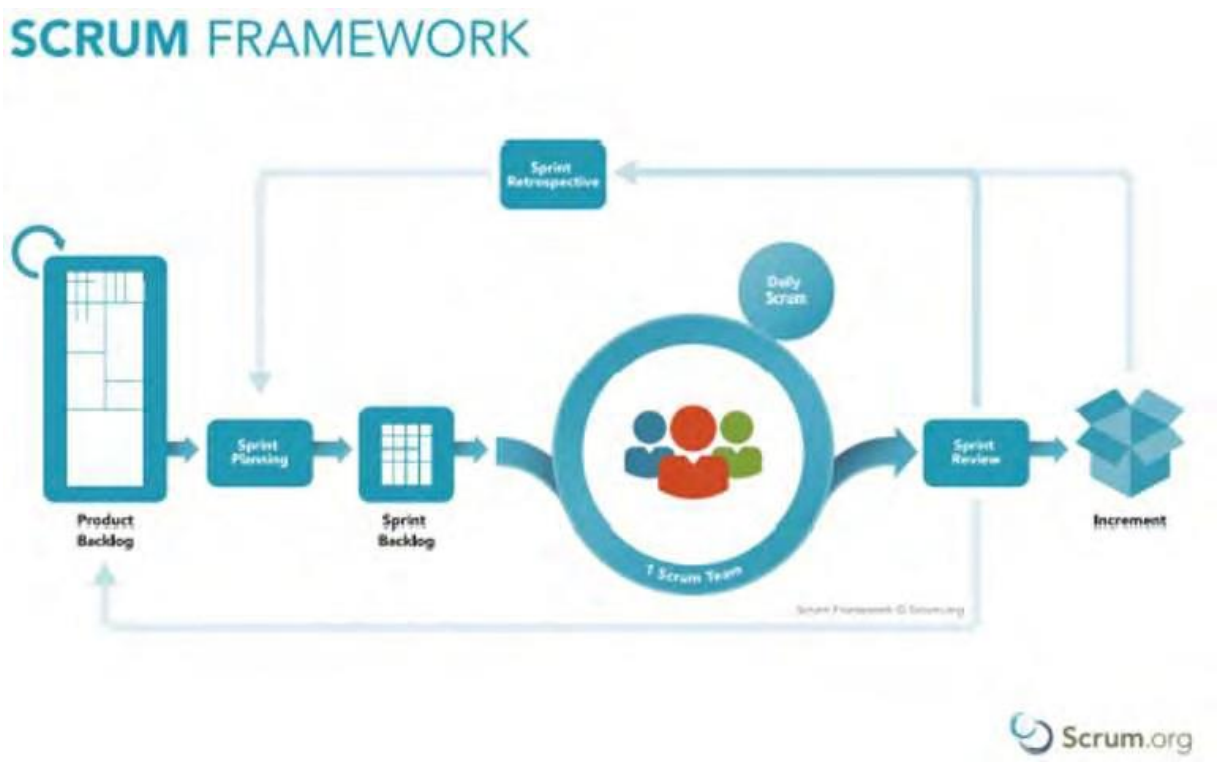


Figure 16 – Scrum Framework by Scrum.org

Scrum Ceremonies

Ceremonies are critical to the success of Scrum. The table below reflects the required ceremonies, including the frequency/timing, participants, and goals of each ceremony. Ceremonies should be held in the appropriate sequence to be effective.

There is significant value to each ceremony, which is magnified when all ceremonies are performed in conjunction with each other.

Agile ceremonies are anticipated to replace traditional project status meetings, ad hoc meetings, routine Project Manager data calls, and status updates.

The duration of each meeting may decrease over time as the team(s) get more comfortable with both facilitation and the goals of each ceremony.

Sequence	Ceremony	Frequency	Timing	Participants	Goals
1	Backlog Grooming	Every 2 weeks (60 minutes)	Just before Kanban/Sprint Planning Meeting	PO, SM, All Team Members	Add, Edit, Delete Issues in the Product Backlog Prioritize "Backlog" Issues to 'To-Do' Estimate the complexity of prioritized items
2	Sprint Planning	Every 2 weeks (60 minutes)	Just before iteration/sprint starts	PO, SM, All Team Members	Determine work that can be completed in the upcoming 2-week work cycle and add to Sprint Backlog
3	Standups	Daily (15 minutes)	During iteration/sprint	SM, All Team Members	SM asks each team member: • What issue(s) did you complete since we last met? • What issue(s) are you currently working on? • Is there anything blocking you?
4	Sprint Demo	Every 2 Weeks (60 minutes)	At end of iteration/sprint	PO, SM, All Team Members, Stakeholder Reps	Team members demo completed work and discuss the increment PO and Stakeholders ask questions, give feedback Work is accepted/rejected by PO
5	Retrospective	Every 2 weeks (60 minutes)	After Sprint Demo	PO, SM, All Team Members	Continuous improvement to enhance team performance(what team should continue doing, stop doing/fix, and what we haven't tried that we should experiment with)

Table 3 – Scrum Ceremonies

The highlighting in the table above shows meetings that may be held back-to-back but they should not be combined. Teams may roll right from a completed Backlog Grooming session into Sprint Planning. Teams may also roll right from a Sprint Demonstration right into a Sprint Retrospective.

For example, if a team chose a two-week Sprint cycle, the meeting schedule may look something like what is shown below. Note: As teams get more efficient, it is not uncommon for meeting durations to shrink.

Figure 17 – Scrum Ceremonies Example

To realize the full value of Scrum ceremonies, teams should:

- Hold all ceremonies on a regular cadence!
- Avoid picking and choosing ceremonies.
- Hold participants accountable for attendance and participation.
- Use standups to identify ownership and blockers only! Do not discuss the status of each story or attempt to solution blockers in the meeting.

Let's Get Started!

You don't need to waste a lot of time making sure everything is perfect to get started.

There will be lots of opportunities to retrospect to improve. Sometimes the best way is to simply pick a product and a team and get moving! Below are some quick steps and a simple roadmap to help you get started at the team level:

- Select a valuable product and perform value stream mapping
- Leverage value stream mapping to identify potential core team members (appropriate Product Owner is critical) and other stakeholders
- Determine an appropriate Agile tool and build backlog, workflow (boards), and metrics
- Get the team trained on both Agile fundamentals and the Agile tool
- Develop a Product Vision (e.g., CNS, SW-ICD), Roadmap, and User Agreement
- Develop Epics/Features for initial release
- Breakdown down Epics/Features for initial release into Stories and gain commitment
- Determine Sprint timebox and schedule Scrum ceremonies accordingly
- Determine what measures to assess for formal feedback in the Value Assessment.
- Get started! Don't worry about being perfect, you can use Retrospectives to improve!
- **If you need some help, visit www.whatisscrum.org**

Figure 18: Getting Started

Key Terms

Although terminology may vary slightly across frameworks, they all contain variations of these key Agile terms:

- **Customer/End-User/User**- Those who will ultimately use the software solution. Users convey operational concepts and requirements/needs, participate in continuous testing activities, and provide feedback on developed capabilities. It is critical for the development team to have a clear understanding who the end-users are, to ensure they are focused on delighting them. A core Agile tenet is active user involvement throughout development.
- **Epic/Feature** – A large body of work to be completed during development. Depending on the Agile framework, the Epic can be too large to complete within a sprint. Features/Epics are further decomposed into smaller user stories. Features/Epics may express business functionality or identify constraints placed on the product or system.
- **Story/User Story** – The smallest unit of requirements written from a user's perspective of how they will use the software. User stories are defined and prioritized by the Product Owner via backlogs. User stories that cannot be completed within a single sprint should be divided into smaller elements. Each user story should have clear acceptance criteria. Depending on the specific Agile framework, some backlogs can contain features and themes, which are additional levels of abstraction of user requirements.
- **Story Points (or Points):** A relative measure of the size or complexity of work required to complete a Story, often using the Fibonacci sequence (1, 2, 3, 5, 8, 13, 21). Story points should not be equated to duration to complete the work and due dates should not be assigned to Stories. Story points are generated by all developers on the team (not individual developers, Product Owner/Scrum Master do not estimate) and represent a team commitment. More complex user stories will require more story points to complete development. Story points should be assigned to each item in the Product Backlog at the end of the Backlog Grooming session. Teams may start by getting aligned on the least complex work and assigning that a point value of 1, then using that relative estimate to guide the Story points for each remaining Story. Story point estimates are unique to each development team.
- **Definition of Done:** A global commitment between the team and Product Owner/Customers/End- Users that applies to all Stories, detailing the activities that must be completed to be considered releasable. Example: All stories must meet Security requirements to be classified as done. All stories must include automated testing to be classified as done.
- **Acceptance Criteria:** Story-specific criteria that the Product Owner/Customer/End-Users require for a Story to be considered completed.
- **Backlog:** A dynamic list of prioritized user stories. Backlogs can be managed for the overall program, each product, and each sprint. The Product Owner is the voice of the customer and own the backlog, updating it based on the feedback of customers/end-users/other stakeholders. The Product Owner regularly grooms the backlog to ensure the work is clearly defined, prioritized, and contains Story Point estimates.
- **Sprint/Iteration/Increment:** Short timeboxes where a defined set of Stories are completed

by the team and releasable to the customer (typically 2-4 weeks with a preference to shorter timeframes). Delivery activities required by the definition of done (plan, design, develop, integrate, test, and demonstrate) occurs within this timebox. The goal is to demonstrate all completed Stories at the end of each timebox to capture fast feedback from the customer that will guide future effort.

- **Release:** Work at the end of a Sprint/Iteration/Increment is releasable, but the Product Owner may determine that a buildup of additional completed Stories is required before the product is released to the customer (typically 1-6 months with a preference to shorter timeframes). Shorter release cycles deliver value faster and result in faster feedback to guide future effort. Timing of release depends upon operational, acquisition, and technical factors that should be discussed with stakeholders across the user and acquisition organizations. As a general guideline, most releases should take less than six months (as championed by US CIO, GAO, and FITARA). Initially, it can be valuable to establish a release cadence to timebox value delivery and identify and resolve challenges, bottlenecks, and impediments in the release process. This helps to avoid inadvertently slipping back into waterfall release durations of a year or more.
- **Velocity:** The measure of the amount of work completed in a given sprint for a given Agile team. Velocity is measured by summing the total number of story points completed by the team. A team's velocity over multiple iterations is a key metric to track a team's performance and aids planning and scheduling future work. Velocity is only applicable to the team and cannot be transferred as an estimation tool for another team. This is because each team measures differently in terms of story points.
- **DevSecOps:** All work related to ensuring the ability to continuously integrate and continuously delivery working code (the pipeline). This responsibility encapsulates multiple areas (i.e., configuration management, automation, development, testing, security, integration, deployment, and operations). It encourages the concept of "shifting-left" as depicted in Figure 6., to reduce handoffs and include all functional areas in planning as early as possible. DevSecOps practices contain their own set of terms and concepts that exceed the scope of this document (e.g., continuous integration; continuous delivery; continuous monitoring; automation; telemetry)
- **Test-Driven Development (TDD):** A practice that involves developing test cases and test scripts before developing the functioning code. Software is then developed and improved upon until it passes the test.
- **Product Roadmap:** A high-level visualization of capability delivery that operates as a strategic plan to guide progression toward the organizational vision. This helps align product owner and stakeholder expectations for future development. Items in the roadmap are value-centric and express "why" items are valuable, but not a detailed explanation not "how". The roadmap is owned by the Product Owner and should be revisited and updated regularly.
- **Minimum Viable Product:** The initial iteration of a new product to customers/end-users that gives them a bare minimum set of functional requirements but enough to respond to in a meaningful way.
- **Story Board / Kanban Boards:** A common Agile tool used in conjunction with a backlog that helps the team track the flow of work from backlog to in progress and through

completion. Helps to ensure ownership or work and minimize work in progress. May include a blocked status to identify bottlenecks/impediments to flow.

- **whatisscrum.org** is a leading provider of professional development and project management solutions. We empower professionals to thrive in today's rapidly evolving business landscape through cutting-edge agile methodologies and industry expertise.