Agile 101
An Agile Primer

Version 1.0

18 November 2019

OUSD(A&S)
This page intentionally left blank.
Table of Contents

1  Purpose 5
2  An Agile Primer 6
   2.1  Comparison to Traditional Waterfall Approach 8
   2.2  Benefits of Agile Delivery 10
   2.3  Common Agile Frameworks 11
   2.4  Common Agile Tools 12
   2.5  Key Agile Terms 12
   2.6  Agile Connection to Lean 14
   2.7  Agile Connection to DevSecOps 15
   2.8  Minimum Viable Product (MVP) Considerations 16
   2.9  Agile Product Roadmap 16
   2.10  Agile Scrum Framework 18
   2.11  Common Agile Roles 19
Appendix A  Abbreviations and Acronyms 22
Table of Figures

Figure 1 - The Agile mindset is comprised of specific values, principles and practices........6
Figure 2 - The Agile Manifesto ....................................................................................................................7
Figure 3 – Comparison of Emergent Design in Iterative Development and Delivery ..............9
Figure 4 - Waterfall vs. Agile software development..................................................................................9
Figure 5 - Agile delivery approach leads to faster value delivery to the user.............................10
Figure 6 - Shifting Left Depiction..............................................................................................................15
Figure 7 - Sample Product Roadmap.........................................................................................................17
Figure 8 -- Sample Agile Product Roadmap............................................................................................17
Figure 9 - Scrum Framework by Scrum.org..............................................................................................18
Figure 10 - Potential Agile Team Construct...............................................................................................20

List of Tables

Table 1 -- Comparison of Traditional vs. Agile Programs .........................................................................10
1 Purpose

The purpose of the Agile 101 Primer is to provide an overview of Agile values, principles, concepts, vocabulary, terms and roles to provide an understanding of the breadth of Agile and how it differs from traditional project management practices.

This is not a “how-to” document and does not provide step-by-step instructions for following a specific framework or methodology such as Scrum or Kanban. Rather, this document provides the foundational information that the reader can use to gain a broad understanding of key and overarching Agile concepts.

This document also touches on Agile ties to Lean and DevSecOps practices.
2  An Agile Primer

Agile is a true paradigm shift in the way work is planned, executed and structured. The structure impacts organizational governance, roles and responsibilities and team structures, among other areas. To improves chances of success in Agile software development it is important to understand the mindset, values and principles that Agile espouses. It starts with the mindset (a Lean mindset) that can be summed up for Agile purposes in 4 values and 12 principles. At execution, practices are applied that align to the Agile mindset, values and principles.

<table>
<thead>
<tr>
<th>AGILE IS A MINDSET</th>
<th>DESCRIBED BY 4 VALUES</th>
<th>DEFINED BY 12 PRINCIPLES</th>
<th>MANIFESTED THROUGH SELECTED PRACTICES</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Brain" /></td>
<td></td>
<td><img src="image" alt="Stars" /></td>
<td>Scrum, Kanban, XP, DevOps, TDD, SAFe, LeSS, ATDD, BDD, Nexus</td>
</tr>
</tbody>
</table>

Agile is based on a set of software development principles that emerged in 2001 after 17 industry leaders created the [Agile Manifesto](https://agilemanifesto.org/) 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 throughout the development cycle. Agile teams must embrace disciplined processes that require training, guidance, and openness to change. The 4 Agile values are as follows:

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan

---

1 [https://agilemanifesto.org/](https://agilemanifesto.org/)
The distinctions made in the 4 value statements do not discount the items on the right. Rather, the items on the left are placed as a greater value to the user because they contribute directly to delivering value faster.\(^\text{3}\) The pace of change is also increasing which requires teams to focus on developing working software incrementally which includes development and delivery of support tools and documentation. The traditional approach to product planning and delivery focused on a predictive approach, whereby planning, requirements and design activities were locked in upfront and included in contracts. The complex nature of projects combined with the increased desire for a faster pace of change has led to dynamics that sometimes become a point of friction between the government and the contractor. Often, the friction is due to mid-execution change requests that impact the scope of work and level of effort. Given this dynamic, if the program team is not confident in the requirements, or if the project duration is longer than a year, programs may want to consider incremental and iterative implementations using Agile approaches.

Agile accepts the nature of large, complex systems and acknowledges that upfront, predictive planning can be challenging. Incremental delivery allows users to receive value faster, and in return, provide feedback faster to the team to be able to apply the learning in the work for the next incremental delivery. The smaller increments (batches) of work and reduced complexity can reduce initial investment costs and improve quality.

The Agile Manifesto also identifies 12 Agile principles:

\(^\text{2}\) Slide source: [https://agilemanifesto.org/](https://agilemanifesto.org/)

\(^\text{3}\) The concept of working on things that contribute to directly to value delivery is a Lean concept that espouses reducing unnecessary work (or waste).
1. Our **highest priority is to satisfy the customer** through **early and continuous delivery** of valuable software.

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

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

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

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

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

7. **Working software** is the primary measure of progress.

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

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

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

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

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

Many of the Agile principles are traceable to Lean principles, which was popularized by Toyota in manufacturing. From the concepts and principles defined in the Agile Manifesto arose a series of Agile frameworks and methodologies used today in software development.

### 2.1 Comparison to Traditional Waterfall Approach

Traditional waterfall projects typically rely upon 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.

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.
As shown in Figure 4, a traditional waterfall development approach takes place in linear phases based on a plan-develop-test-deploy cycle. User feedback is generally not received until the full solution is developed, and the end-user receives no value for years. An Agile development lifecycle promotes continuous delivery, which provides opportunities for user feedback within each delivery cycle. This iterative cycle can provide incremental value in much quicker timespans as opposed to the traditional waterfall approach.

---

4 https://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp
5 Source: Web Applications and Data Integration Company
Table 1 highlights some of the differences between traditional and Agile programs.

**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.

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
<th>Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mindset</strong></td>
<td>Define rigid requirements; design, develop, produce</td>
<td>Collaborative culture to iteratively deliver priority capabilities to users</td>
</tr>
<tr>
<td><strong>Size/Scope</strong></td>
<td>Multi-year releases</td>
<td>&lt;6 month releases</td>
</tr>
<tr>
<td><strong>Requirements</strong></td>
<td>Defined up front via large CDDs and contracts</td>
<td>Iteratively defined and continuously prioritized via dynamic backlogs</td>
</tr>
<tr>
<td><strong>Cost Estimate</strong></td>
<td>Exhaustive upfront analysis; rigid baselines</td>
<td>Iterative; integrated; collaborative through development cycles</td>
</tr>
<tr>
<td><strong>Testing</strong></td>
<td>Long timelines following system development</td>
<td>Automated, daily, integrated throughout development</td>
</tr>
</tbody>
</table>

### 2.2 Benefits of Agile Delivery

There are several key benefits to Agile:

- **Faster delivery of value** – By delivering work in smaller increments, the work can be completed and delivered faster to the user as indicated by Figure 5.

![Agile Delivery Approach](image)  

*Figure 5 - Agile delivery approach leads to faster value delivery to the user*

- **More opportunities for feedback** – The incremental delivery of value to the user translates to more opportunities to receive feedback, fine-tune requirements and
adjust without overcommitting to very large batches of work. In the traditional approach, the user might not have an opportunity to gauge the value of the delivered work until all the requirements are delivered.

- Reduced risk – The smaller batches of work (or limited the amount of work being performed at any given time) naturally reduces the complexity and cost of work being performed. If the work turns out to be of little to no value, the team can work together with the users to quickly adjust or the work can be stopped with limited sunk cost relative to a fully scoped waterfall project.
- Improved overall collaboration – The team structure provides more opportunities for collaboration with users and other team members.
- Improved quality – The incremental nature of the work, combined with some of the Lean and DevSecOps practices described in this document will lead to improved quality of work over time. There are many reasons for the improved quality, but some of the main reasons result from smaller batches of work delivered incrementally; product/goal-oriented team structures as opposed to siloed, functionality-oriented team structures; closer collaboration among the requirements, development, testing, security and operations units; and assignment of accountability and decision-making to where the information and expertise resides.

2.3 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:

- **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.\(^6\)

- **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.\(^8\)

- **Scaled Agile Framework (SAFe)** – A framework intended mostly for larger projects and programs that is based on Lean-Agile principles and addresses five core

---

\(^6\) The Kanban board concept originated from the 1978 book by Taiichi Ohno called “Toyota Production System – Beyond Large-Scale Production”, which described LEAN, Lean manufacturing and Kanban.

\(^7\) [https://www.agilealliance.org/glossary/xp/](https://www.agilealliance.org/glossary/xp/)

\(^8\) [http://www.extremeprogramming.org/](http://www.extremeprogramming.org/)
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 include:
- Large-Scale Scrum (LeSS)
- Dynamic Systems Development Method (DSDM),
- Disciplined Agile Delivery (DaD),
- Nexus Framework by Scrum.org

### 2.4 Common Agile Tools

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

- **Backlog** – A ledger of Agile requirements written in the form of mostly user stories, and epics. Types of backlogs include the Product Backlog, Increment or Release Backlog and Sprint Backlog. Each type of Backlog contains the Agile requirements for the given product, increment, release or sprint respectively.
- **Kanban Board**: A tool used to track the flow of work and make work visible by showing work in queue and work in progress. Depending on the team, the board can also show blocked work and work dependencies.

### 2.5 Key Agile Terms

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

- **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** – 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. Epics are further decomposed into smaller features and user stories. Epics may express business functionality or identify constraints placed on the product or system.
- **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.

---

9 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 unit of measure to determine the size or amount of work required by a development team to complete a user story. More complex user stories will require more story points to complete development. Most Agile teams work together with the Product Owner to estimate the story points for each user story in the Product Backlog. Usually, story points are measured as numeric numbers, for example, a story point of 1 is the smallest unit and all other work is assessed relative to that size, but other sizing techniques are also used. Story points are unique to each development team.

• **Definition of Done** – A shared and understood definition of the activities that must be completed for a given user story to be considered complete. The definition of done is used to provide clarity around the expected quality of work to meet the users’ needs.

• **Backlog** - A dynamic list of prioritized user stories. Backlogs can be managed for the overall program, each product, and each sprint. The Product Owner, working with stakeholders, will regularly groom the backlogs to ensure the work is clearly defined, in priority order, and with reasonable estimates. The higher priority work will have the greatest details defined and completed first, whereas work planned for future releases will be lower on the backlogs with less fidelity.

• **Sprint** - A short cycle of work (notionally two to four weeks in duration) that focuses on completing a defined subset of project deliverables, or usable functionality. Each sprint includes planning, design, development, integration, testing, and demonstrating working software to the product owner, users, and other stakeholders.

• **Release** - The release represents the core element of the program structure, guiding how frequently the program delivers capabilities to the end users. The length of each 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). Shorter release cycles have several benefits, the most important being that the program deploys useful capability to the end-user faster.

• **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 deliver working code. 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. The software is then developed and improved upon until it passes the test.

- **Product Roadmap**: A high-level strategic plan to guide organizational vision and align product owner and stakeholder expectations for future development. The roadmap, like the backlog, should be revisited and updated regularly.

- **Minimum Viable Product** – A concept adopted from Lean that emphasizes using a scientific, measured and incremental approach to delivering value, which starts by thinking of value delivery as a series of scientific experiments. In Agile, the term has a more specific definition beyond the concept, which will be discussed later in this document.

- **Kanban** – A framework used by Agile teams to make work visible. The most commonly recognized tool within the Kanban framework is the Kanban Board, which is used to track the flow of work and make work visible by showing work in queue and work in progress. The Kanban Board can also show blocked work and work dependencies.

### 2.6 Agile Connection to Lean

Agile and DevSecOps practices are based on Lean principles of

- Delivering value quickly – By managing flow of work in order 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.

- Building-in quality at all levels of work – By ensuring accountability at all levels of work. The team needs to avoid single point quality checks in favor of quality checks at all levels of work, which involves everyone on the team from developers to testers, security and operations team members. 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)

- Eliminating waste – Reducing “gold plating” or overbuilding based on assumptions of desired user value; delivering in small batches to reduce complexity and possible rework resulting from issues that naturally rise from complexity; Adopting an MVP design to begin to gain valuable feedback for incremental improvements.

- Making decisions at the last responsible moment – This really applies to the idea that the team should avoid locking down a design early and let the data and feedback determine design updates

- Taking a scientific approach – Which includes making data driven decisions based on testing specific hypothesis and iterating based on the learnings. Examples of this include application of MVP concept to be able to test smaller pieces of functionality; experimentation and increased collaboration with users through application of User-Centered Design concepts to test usability before implementing.
• Creating a learning culture based on respect – A culture that promotes continuous learning, improvement and innovation, and encourages a blameless culture that allows teams and individuals to take calculated risks and learn openly.

2.7 Agile Connection to DevSecOps

Agile focuses more on the organizational structure, planning and organizing work to be completed. DevSecOps provides the architecture path (or pipeline) for delivering working software at a pace that can keep up with the pace of the Agile 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. In other words, DevSecOps looks at all the key steps involved to deliver the work from initial request to operations and beyond.

By taking an end-to-end value delivery approach, DevSecOps encourages the development, testing, security and operations teams to work together as early and often as possible in the application life cycle. The concept of working together and collaborating as early as possible is captured in Figure 6.

Figure 6 - Shifting Left Depiction

DevSecOps encourages use of automation and configuration control to make activities like testing, integration and deployment easily repeatable by implementing automated tests along the delivery pipeline, taking advantage of automated deployment to consistent environments that are instantiated with the use of automated code, configuration and other cloud technologies such as containerization, orchestration and as noted, automation.
2.8 Minimum Viable Product (MVP) Considerations

The MVP is the first delivery of value to the user, which begins the process of continuous value delivery while reducing risk from large batch size deliveries. Future iterations after deployment/delivery of the MVP add functionality/capability to mature the solution and provide a rich experience. 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, demonstrable set of capabilities. The MVP feedback received from end-users is critical to help decide the future of the solution. 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. The feedback from users can also identify that the product/service isn’t needed, negating the need for future iterations.

2.9 Agile Product Roadmap

Traditional projects using a waterfall methodology usually require a detailed Integrated Master Schedule (IMS) to develop, test, integrate and deliver a product. The schedule assumes that the team has confidence in the activities through final product delivery and generally expects minimal variance in schedule or cost. If the schedule changes significantly (usually exceeding 10%) as a result of requirements, design, or scope changes, the team must engage in a time-consuming replanning effort to update the IMS. Changing the IMS has downstream impacts on resources and cost as the schedule usually shifts to the right. Resulting contract changes and the overall impact on the current work are high.

---

The Agile roadmap provides a prioritized, value-driven view of work to be performed that also aligns to the program vision. The roadmap depicts the capabilities or features to be developed over time while avoiding overly prescriptive plans. Because Agile roadmaps are defined at a higher level of abstraction, they allow the teams to collaborate on lower level requirements and design details.

---

From GSA Tech Guides https://tech.gsa.gov/guides/develop_an_agile_product_roadmap/

---

From GSA Tech Guides https://tech.gsa.gov/guides/develop_an_agile_product_roadmap/
The initial version of the roadmap should, at a minimum, depict the capabilities and/or features to be delivered for the MVP (organized by iteration) and align to the overall product vision.

To avoid recreating an integrated master schedule (IMS), the Program should define the roadmap at a high enough level that shows the desired direction of the program and what it wants to achieve in terms of value (described as capabilities, or perhaps features). The roadmap should not be task or activity oriented, but outcome/goal oriented, so that it leaves enough room for the team to innovate and learn along the way.

2.10 Agile Scrum Framework

The Agile Scrum framework, one of the most popular frameworks for Agile teams, is depicted in Figure 9. The framework is also used in its entirety, or as a derivative, in other frameworks. The Scrum framework is also a very common first step for organizations looking to move into Agile project management.

The framework is designed to deliver incremental capabilities in 2-4 week sprints. Each sprint contains a series of events, called ceremonies for each sprint.

- **Sprint Planning** – Involves the sprint team working together to define the scope of the Sprint. The Sprint Backlog defines the scope of work for the sprint, and is a subset of items from the Product Backlog (the master set of Agile requirements for the product). The team must commit to the scope of work for each sprint. The team can use the velocity from previously completed sprints to estimate their sprint capacity.

- **Sprint Execution (including the Daily Scrum)** – Once the team commits to the sprint, the sprint can begin. The team holds daily Scrums, facilitated by the Scrum Master. Each Scrum is 15-20 minutes long. Each team member that is assigned work provides an update on their progress, their plan moving forward, and any blockers or roadblocks that is preventing them from accomplishing their work.
● **Sprint Review (also known as Sprint Demonstration)** – At the end of each sprint, the team demonstrates the completed work to the Product Owner and selected users. The team collects feedback on the work and documents new work in the Product Backlog resulting from the feedback.

● **Sprint Retrospective** – In the spirit of continual learning, the team reviews the sprint to discuss lessons learned and adjust as needed.

**Important Note**: Some level of upfront planning is required before a team can start the first sprint. The team must identify enough of the Agile requirements in the Product Backlog to be able to start. Some teams also define their Minimum Viable Product (MVP), which is the minimum scope required to release as the first release. The MVP should contain enough value to the users to be able to collect meaningful feedback from users. Additionally, teams define an initial product roadmap that is functionally oriented as opposed to task oriented. The product roadmap helps the team set a pace for work and provides just enough information to plan work for future sprints.

### 2.11 Common Agile Roles

● **Agile Team** - Agile development requires both the contractor development team and the Government program office to adopt a new set of roles and responsibilities. Figure 10 shows a potential Agile team construct. At the heart of an Agile software development effort is the release team responsible for execution. The release team includes a core team composed of the project manager, product owner, tester, and systems engineer, and a development team led by a scrum master (scrum master and Product Owner are roles specific to the Scrum methodology, team roles may vary if using other Agile methods). The broader extended team includes 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.
- **Scrum Master** *(included in Agile Scrum and SAFe frameworks, among others)* – The Scrum Master is focused on ensuring the execution of good Scrum and championing at process to ensure delivery of value to the customer. The Scrum Master also leads the daily Scrums and helps resolve issues or remove blockers so that the team can continue to perform well during the sprint. This is typically a member of a contractor team, depending on the details of the individual contract.

- **Product Owner** – The authoritative user community representative who represents the business by maintaining accountability for the Product Backlog, ensuring that the priorities of the business are represented in the work being performed, communicates the acceptance criteria and accepts delivery of work. 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.

- **DevSecOps Engineer or Lead** – Responsible for all or part of the continuous integration and continuous delivery activities to ensure that there are no technical barriers to developing, securing, integration, testing and delivery of working code or product.

- **Development Team** – The cross-functional team (usually 5-12 depending on the framework) typically composed of contractor staff that includes representation across functional areas (i.e., requirements, engineering/design, development, security, testing, operations) that work together to deliver the intended product value to the customer.

---

13 https://aida.mitre.org/models/agile-acquisition/agile-overview/
- **Development Team Lead** – Represents the development team when needed to negotiate scope of work for a given release or sprint.

- **Agile Release Train (SAFe only)** – The Agile Release Train (ART) is comprised of a team of Agile teams that are responsible for delivering a specific value (usually a product).

- **Release Train Engineer (SAFe only)** – The Release Train Engineer (RTE) in SAFe is the servant leader, coach and facilitator for the ART. One of the primary responsibilities of the RTE is to plan and orchestrate the Program Increment (PI) Planning events, which are significant milestone ceremonies in SAFe.
## Appendix A  Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoD</td>
<td>Department of Defense</td>
</tr>
<tr>
<td>FISMA</td>
<td>Federal Information Security Management Act</td>
</tr>
<tr>
<td>GAO</td>
<td>Government Accountability Office</td>
</tr>
<tr>
<td>GSA</td>
<td>General Services Administration</td>
</tr>
<tr>
<td>IMS</td>
<td>Integrated Master Schedule</td>
</tr>
<tr>
<td>MVP</td>
<td>Minimum Viable Product</td>
</tr>
<tr>
<td>OUSD</td>
<td>Office of the Undersecretary of Defense</td>
</tr>
<tr>
<td>PM</td>
<td>Program/Project Manager</td>
</tr>
<tr>
<td>PO</td>
<td>Product Owner</td>
</tr>
<tr>
<td>SAFe</td>
<td>Scaled Agile Framework</td>
</tr>
<tr>
<td>TDD</td>
<td>Test-Driven Development</td>
</tr>
</tbody>
</table>