USGIF NRO Industry Advisory Group
Agile DevSecOps Terms of Reference

	Synonyms	SAFe® https://www.scaledagileframework.co	Agile Alliance® https://www.agilealliance.org/agile101/agile-	SmartSheet https://www.smartsheet.com/ultimate-agile-	18F Lexicon https://agile.18f.gov/agile-lexicon/	Techtarget "WhatIs" Glossary https://whatis.techtarget.com/	Red Hat https://www.redhat.com/en/topics/devops/what-
Agile Software Development				Agile Software Development refers to the project management approach of developing increments of software in frequent iterations based on evolving requirements.	pertaining to the Agile Manifesto and/or Agile Principles (http://agilemanifesto.org). Agile is a set of values and principles that describe a way of working that promotes continuous learning and user-focused value delivery.	In software application development, Agile is a methodology that **anticipates the need for flexibility and applies a level of pragmatism into the delivery of the finished product**. Agile requires a **cultural shift** in many companies because it focuses on the clean delivery of individual pieces or parts of the software and not on the entire application.	
Backlog Grooming/Refinement			Backlog grooming is when the product owner and some, or all, of the rest of the team refine the backlog on a regular basis to ensure the backlog contains the appropriate items, that they are prioritized, and that the items at the top of the backlog are ready for delivery.	Backlog grooming occurs at the end of a sprint, when the team meets to make sure the backlog is ready for the next sprint. The team may remove user stories that aren't relevant, create new stories, reassess priority, or split user stories into smaller tasks. Backlog grooming is both an ongoing process and the name for the meeting where this action occurs (a backlog grooming meeting).		Product backlog grooming, also called product backlog refining, is an Agile software development process in which the development team revisits a product backlog that has been pre-defined by the team's facilitator. A backlog is a set of tasks that must be finished before code can be released. During a product backlog grooming session, the development team works with business owners to prioritize a list of work that is backlogged and break the list into user stories for future use.	
Behavior Driven Development			BDD is a practice where members of the team discuss the expected behavior of a system in order to build a shared understanding of expected functionality.			Behavior-driven development (BDD) is an Agile software development methodology in which an application is documented and designed around the behavior a user expects to experience when interacting with it. By encouraging developers to focus only on the requested behaviors of an app or program, BDD helps to avoid bloat, excessive code, unnecessary features or lack of focus. This methodology combines, augments and refines the practices used in test-driven development (TDD) and acceptance testing.	
Continuous Improvement	Kaizen				Continuous improvement is a process of improving quality and efficiency by making **small, incremental changes over time**. In Kanban, continuous improvement refers specifically to the process of optimizing workflow and reducing cycle time, resulting in increased productivity.	Kaizen is an approach to creating continuous improvement based on the idea that small, ongoing positive changes can reap major improvements. Typically, it is based on **cooperation and commitment and stands in contrast to approaches that use radical changes or top-down edicts to achieve transformation.** Kaizen is core to lean manufacturing, or The Toyota Way. It was developed in the manufacturing sector to lower defects, eliminate waste, boost productivity, encourage worker purpose and accountability, and promote innovation.	
Continuous Integration		Continuous Integration is the process of taking features from the Program Backlog and developing, testing, integrating, and validating them in a staging environment where they are ready for deployment and release	Teams practicing continuous integration seek two objectives: minimize the duration and effort required by each integration episode, be able to deliver a product version suitable for release at any moment. In practice, this dual objective requires an integration procedure which is reproducible at the very least, and largely automated. This is achieved through version control tools, team policies and conventions, and tools specifically designed to help achieve continuous integration.		Continuous integration is a software engineering practice that involves continual integration of new development code into the existing codebase.	Continuous integration (CI) is a software engineering practice in which **frequent, isolated changes are immediately tested** and reported on when they are added to a larger code base.	
Continuous Delivery						Continuous delivery (CD) is a software release approach in which development teams produce and test code in short cycles, usually with a high degree of automation. This process enables development teams to build, test and deploy software quickly by **encouraging more incremental updates, rather than a complete overhaul of a given product.**	
Continuous Deployment		Continuous Deployment is the process that takes validated features from a staging environment and deploys them into the production environment, where they are readied for release	Continuous deployment can be thought of as an extension of continuous integration, aiming at minimizing lead time, the time elapsed between development writing one new line of code and this new code being used by live users, in production.			Continuous deployment is a strategy for software releases wherein any code commit that passes the automated testing phase is automatically released into the production environment, making changes that are visible to the software's users. Continuous deployment eliminates the human safeguards against unproven code in live software. It should only be implemented when the development and IT teams rigorously adhere to production-ready development practices and thorough testing, and when they apply sophisticated, real-time monitoring in production to discover any issues with new releases.	
Definition of Done			The definition of done is an agreed upon list of the activities deemed necessary to get a product increment, usually represented by a user story, to a done state by the end of a sprint.	Definition of Done refers to a set of predetermined criteria that a product needs to meet in order to be considered as being done. The team reaches a consensus as to what defines a task as being done and then posts a checklist of steps that must be completed before the product can be considered as potentially shippable. The team displays this list in the form of a Big Visual Chart prominently in the team area.	A working agreement among the team detailing the standard for achieving a "Done" Product Backlog Item (i.e, user story). This applies to all user stories and includes meeting acceptance criteria.	A definition of done is a checklist of criteria that a product, product increment or project must satisfy to be considered completed.	
DevOps		DevOps is a mindset, a culture, and a set of technical practives. It provides communication, integration, automation, and close cooperation among all the people needed to plan, develop, test, deploy, release, and maintiain a Solution				DevOps is the blending of the terms development and operations, meant to represent a collaborative or shared approach to the tasks performed by a company's application development and IT operations teams. The term DevOps is used in several ways. In its most broad meaning, DevOps is an operational philosophy that promotes better communication between these teams -- and others. In its **most narrow interpretation, DevOps describes the adoption of automation** and programmable software development and infrastructure deployment and maintenance.	

USGIF NRO Industry Advisory Group
Agile DevSecOps Terms of Reference

DevOps 2.0				DevOps 2.0 is the extension of DevOps practices through the entire organization, beyond development and IT ops. DevOps 2.0 aims to break down silos and foster communication and cooperation between all groups -- technical and non-technical -- involved in the conception, production and maintenance of software. A DevOps 2.0 process includes development and operations, quality assurance and security, HR and legal, and on through the organization.	
DevSecOps					DevSecOps means thinking about application and infrastructure security from the start. It also means automating some security gates to keep the DevOps workflow from slowing down.
SecOps				SecOps is a management approach that connects security and operations teams, similar to how DevOps unifies software developers and operations professionals to increase efficiency and eliminate priority conflicts.	
Epic	An Epic is a container for a Solution development initiative large enough to require analysis, the definition of a Minimum Viable Product (MVP), and financial approval before implementation.	An epic is a large user story.	Epic or epic stories are defined as large user stories that, in their current state, would be difficult to estimate or to complete in a single iteration. Epic stories are typically lower priority and are waiting be broken down into smaller components.		
Feature	A Feature is a service that fulfulls a stakeholder need. Each feature includes a benefit hypothesis and acceptance criteria, and is sized or split as necessary to be delivered by a single Agile Release Train (ART) in a Program Increment (PI)				
Framework				In computer systems, a framework is often a layered structure indicating what kind of programs can or should be built and how they would interrelate. Some computer system frameworks also include actual programs, specify programming interfaces, or offer programming tools for using the frameworks. A framework may be for a set of functions within a system and how they interrelate; the layers of an operating system; the layers of an application subsystem; how communication should be standardized at some level of a network; and so forth. A framework is generally more comprehensive than a protocol and more prescriptive than a structure.	
Fail-Fast			Fail-fast is the process of starting work on a task or project, obtaining immediate feedback, and then determining whether to continue working on that task or take a different approach—that is, adapt. If a project is not working, it is best to determine that early on in the process rather than waiting until too much money and time has invested.	Fail fast is a philosophy that values extensive testing and incremental development to determine whether an idea has value. An important goal of the philosophy is to **cut losses when testing reveals something isn't working and quickly try something else**, a concept known as **pivoting**. Fail fast is often associated with the lean startup methodology. The philosophy, which is aligned with management by objectives, is often embraced by businesses that want to develop new products and services with less financial risk than traditional approaches to product development provide. The concept of failing fast is also associated with differences between the waterfall and agile approaches to software development.	
Feedback					
Flow	The key to successfully executing SAFe is to establish a continuous flow of work that supports incremental value delivery, based on constant feedback and adjustment. Continuous flow enables faster value delivery, effective Built-In Quality practices, relentless improvement, and evidence-based governance.			Flow, in the context of psychology, is a state of intense engagement, focus and contentment in the present moment and current activity. Sometimes referred to as being "in the zone," flow states are known to enhance creativity and performance and spark innovation. The experiences are considered profound enough to improve the individual's overall satisfaction in life.	

Lean Software Development						Lean Software Development is an example of lightweight Agile methodology applied to project development. Lean Software Development combines the Lean manufacturing approach pioneered by Toyota in the 1950s (also known as just-in-time production) and Lean IT principles, and applies them to software. LSD places a strong emphasis on people and effective communication.
Metrics		Metrics are agreed-upon measures used to evaluate how well the organization is progressing toward the portfolio, large solution, program, and team's business and technical objectives.				In software development, a metric (noun) is the measurement of a particular characteristic of a program's performance or efficiency.
Microservice						Microservices, or microservice architecture, is an approach to application development in which a large application is built as a suite of modular components or services. Each module supports a specific task or business goal and uses a simple, well-defined interface, such as an application programming interface (API), to communicate with other sets of services. Software developer and author Martin Fowler is credited with promoting the idea of breaking down services in a service-oriented architecture (SOA) into microservices.
Minimal Viable Product			A Minimum Viable Product is, as Eric Ries said, the "version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort."		The smallest usable thing you need to start validating the actual idea of the business or product. The very first increment of the build/measure/learn cycle (Zappos example in The Lean Startup.) It's there to help demonstrate the market exists for a particular idea. The term "MVP" means something different to virtually everyone, so be sure to get clarity from the person or persons using the term in conversations.	Minimum viable product (MVP) is a concept for making a product that fills the perceived needs of a customer or solves a problem adequately enough to expect a sustainable business around it. An MVP attempts to strike the right balance in a product so that it meets expectations without expending effort on things customers don't care about.
Planning Poker			An approach to estimation used by Agile teams. Each team member "plays" a card bearing a numerical value corresponding to a point estimation for a user story.	Planning Poker is a team building exercise or game used to arrive at a group consensus for estimating workload.		Planning Poker is a teambuilding activity for achieving group consensus. It is used by agile software development teams to estimate how long a component of a software project (story) will take to complete.
Platform						In IT, a platform is any hardware or software used to host an application or service. An application platform, for example, consists of hardware, an operating system and coordinating programs that use the instruction set for a particular processor or microprocessor. In this case, the platform creates a foundation that ensures object code will execute successfully.
Program/Product Manager Product Owner	*		The product owner is a role created by the Scrum Framework responsible for making sure the team delivers the desired outcome	As a member of the Agile team, the Product Owner represents the customer, and conveys the customer's requirements and vision to the team. The product owner writes the acceptance criteria, and prioritizes and maintains the product backlog. Product owners should be able to communicate well in both directions: both taking team concerns to the customer and stakeholders, and ensuring that the team stays on track to meet the customer's vision for the product.	Product Owner owns the product vision and incorporates stakeholder feedback to set priorities and manage the backlog	Product owner is a scrum development role for a person who represents the business or user community and is responsible for working with the user group to determine what features will be in the product release.
Program Backlog	Product Backlog	Holding area for upcoming features, which are intended to address user needs and deliver business benefits for a single Agile Release Train (ART). It also contains the enabler features necessary to build the Architectural Runway	A product backlog is a list of the new features, changes to existing features, bug fixes, infrastructure changes or other activities that a team may deliver in order to achieve a specific outcome.			A backlog is a set of tasks that must be finished before code can be released.
Relative Estimation			Relative estimation consists of estimating tasks or user stories by comparison or by grouping of items of equivalent difficulty.			
Release	Program Increment SPIN Phase	A Program Increment (PI) is a timebox during which an Agile Release Train (ART) delivers incremental value in the form of working, tested software and systems. PIs are typically 8-12 weeks long. The most common pattern for a PI is four development iterations, followed by one innovation and planning (IP) iteration.		An Agile release refers to the final delivery of a software package after the completion of multiple iterations or sprints. A release can be either the initial build of an application or the addition of one or more features to an existing application. A release should take less than a year to complete, and in some cases, may only take three months.	a usable increment of customer value delivered to users in the wild.	A release is the distribution of the final version of an application. A software release may be either public or private and generally constitutes the initial generation of a new or upgraded application. A release is preceded by the distribution of alpha and then beta versions of the software. In agile software development, a release is a deployable software package that is the culmination of several iterations. Releases can be made before the end of an iteration.

USGIF NRO Industry Advisory Group
Agile DevSecOps Terms of Reference

Scrum			Scrum is a process framework used to manage product development and other knowledge work.	Scrum is the most widely used framework under the Agile umbrella. Scrum is an iterative software model that follows a set of predefined roles, responsibilities, and meetings. In Scrum, iterations are called sprints and are assigned a fixed length—sprints typically last one to two weeks, but can last as long a month.	a framework for Agile product development centered around self-organizing team, customer focus, and responding to change.	Scrum is a framework for project management that emphasizes teamwork, accountability and iterative progress toward a well-defined goal. The framework begins with a simple premise: Start with what can be seen or known. After that, track the progress and tweak as necessary. The three pillars of Scrum are transparency, inspection and adaptation.
Scrum Meeting			The daily meeting is one of the most commonly practiced Agile techniques and presents opportunity for a team to get together on a regular basis to coordinate their activities.	The Daily Scrum is a brief communication and status-check session facilitated by the Scrum Master where Scrum teams share progress, report impediments, and make commitments for the current iteration or sprint. The Daily Scrum consists of a tightly focused conversation kept to a strict timeframe; the meeting is held at the same time, every day (ideally, in the morning), and in the same location. The Scrum task board serves as the focal point of the meeting.	A short, daily meeting during which the team self-organizes for the day. Never more than 15 minutes. Information exchanged can vary but always covers what each team member did yesterday, what they plan to do today, and what if anything is blocking their work. Retrospective - A meeting of the whole team held once per sprint, wherein the team reflects on what happened in the iteration and identifies actions for team improvement going forward. This is a critical part of the Agile continuous improvement, as contrasted with a post-mortem which happens only after launch, which limits the opportunity for applying lessons learned.	The Daily Scrum is a short stand-up meeting that happens at the same place and time each day. At each meeting, the team reviews work that was completed the previous day and plans what work will be done in the next 24 hours. This is the time for team members to speak up about any problems that might prevent project completion.
Scrum Team	Agile Team	The SAFe Agile Team is a cross-functional group of 5 to 11 people who have the responsibility to define, build, test, and where applicable deploy, some element of Solution value—all in a short Iteration timebox. Specifically, the SAFe Agile Team incorporates the Dev Team, Scrum Master, and Product Owner roles	A "team" in the Agile sense is a small group of people, assigned to the same project or effort, nearly all of them on a full-time basis.	A Scrum team typically comprises five to nine members with cross-functional skills. Unlike traditional teams of developers, there are no specific roles. A Scrum team is self-organized and self-contained—the team should have the right amount of members with the appropriate skills needed to complete the sprint.		
Scrum Master		The ScrumMasters are servant leaders and coaches for an Agile Team. The help educate the team in Scrum, Extreme Programming (XP), Kanban, and SAFe, ensuring that the agreed Agile process is being followed. They also help remove impediments and foster an environment for high-performing team dynamics, continuous flow, and relentless improvement.	The scrum master is responsible for ensuring the team lives agile values and principles and follows the practices that the team agreed they would use.	The Scrum Master is often viewed as the coach for the team. He or she organizes meetings, resolves roadblocks and issues, and works with the product owner to make sure the product backlog is up to date. The Scrum Master does not have any authority over team members, however, he or she does have authority over the process. A Scrum Master may complete formal training to become a certified Scrum Master but this is not required.	the team member who is responsible for the team's rituals and process, who ensures the team can execute on its commitments by removing blockers and facilitating activities like the daily standup, sprint planning, and retrospectives.	This team member serves as a facilitator. The Scrum master is responsible for ensuring that Scrum best practices are carried out and the project is able to move forward.
Scrum of Scrums				A Scrum of Scrums meeting is a scaling mechanism used to manage large projects involving Scrum multiple teams. A Scrum of Scrums is held to facilitate communication between teams that may have dependencies on one another. One member from each team attends the Scrum of Scrums to speak for the team—this could be the Scrum Master but may be any team member who can effectively relay information and handle questions or concerns for the team.		
Sprint	Iteration	Iteration - Iterations are the basic building block of Agile development. Each iteration is a standard, fixed-length timebox, where Agile Team deliver incremental value in the form of working, tested software and systems. The recommended duration is two weeks. However, one to four weeks is acceptable, depending on the business context.	An iteration is a timebox during which development takes place. The duration may vary from project to project and is usually fixed.	An iteration is a fixed or timeboxed period of time, generally spanning two to four weeks, during which an Agile team develops a deliverable, potentially shippable product. A typical Agile project consists of a series of iterations, along with a planning meeting prior to development and a retrospective meeting at the end of the iteration. Iterations are referred to as sprints in Scrum.	the unit of iteration in scrum: a short, fixed interval in which an agile team commits to and delivers a piece of customer value from the backlog. Sprint Backlog Item - A more granular item than one one would find in a product backlog. It could be a task, chore, bug, or small or finely sliced user story. These items are usually what you find in a trello or waffle board.	In agile software development, an iteration is a single development cycle, usually measured as one week or two weeks. An iteration may also be defined as the elapsed time between iteration planning sessions. While the adjective iterative can be used to describe any repetitive process, it is often applied to any heuristic planning and development process where a desired outcome, like a software application, is created in small sections. These sections are iterations. Each iteration is reviewed and critiqued by the software team and potential end-users; insights gained from the critique of an iteration are used to determine the next step in development. Data models or sequence diagrams, which are often used to map out iterations, keep track of what has been tried, approved, or discarded -- and eventually serve as a kind of blueprint for the final product.
Sprint Backlog			A sprint backlog is the subset of product backlog that a team targets to deliver during a sprint in order to accomplish the sprint goal and make progress toward a desired outcome.	A sprint backlog is a segment of Product Backlog Items (PBIs) that the team selects to complete during a Scrum sprint. These PBIs are typically user stories taken from the product backlog.		
Sprint Goals		Iteration Goals are a high-level summary of the business and technical goals that the Agile Team agrees to accomplish in an iteration. They are vital to coordinating an Agile Release Train (ART) as self-organizing, self-managing team of teams		A sprint backlog is a segment of Product Backlog Items (PBIs) that the team selects to complete during a Scrum sprint. These PBIs are typically user stories taken from the product backlog.		

USGIF NRO Industry Advisory Group
Agile DevSecOps Terms of Reference

Sprint Planning		Iteration Planning is an event where all team members determine how much of the Team Backlog they can commit to delivering during an upcoming iteration. The team summarizes the work as a set of commited Iteration Goals.	Sprint planning is an event that occurs at the beginning of a sprint where the team determines the product backlog items they will work on during that sprint.	The Sprint Planning Meeting is a working session held before the start of each sprint to reach a mutual consensus between the product owner's acceptance criteria and the amount of work the development team can realistically accomplish by the end of the sprint. The length of the sprint determines the length of the planning meeting, with two hours being equivalent to one week of the sprint. Using this formula, the planning meeting for a two-week sprint would last about four hours, although this can vary.		Each sprint begins with a planning meeting. During the meeting, the product owner (the person requesting the work) and the development team agree upon exactly what work will be accomplished during the sprint. The development team has the final say when it comes to determining how much work can realistically be accomplished during the sprint, and the product owner has the final say on what criteria need to be met for the work to be approved and accepted.
Sprint Retrospective		Iteration Retrospective is a regular meeting where Agile Team members discuss the results of the iteration, review their practices, and identify ways to improve		A Scrum Retrospective is a meeting held following the completion of a sprint to discuss whether the sprint was successful and to identify improvements to be incorporated into the next sprint.		An Agile retrospective is a meeting that's held at the end of an iteration in Agile software development (ASD). During the retrospective, the team reflects on what happened in the iteration and identifies actions for improvement going forward.
Sprint Review		The Iteration Review is a cadence-based event, where each team inspects the increment at the end of every iteration to assess progress, and then adjusts its backlog for the next iteration.		The Scrum team holds a Sprint Review meeting immediately following the completion of a sprint to review and demonstrate what the team has accomplished during the sprint. This meeting is attended by the product owner or customer, Scrum Master, Scrum team, and stakeholders. The Sprint Review is an informal meeting (no Powerpoint slides allowed). The length of the sprint determines the length of the review meeting, with one hour being equivalent to one week of the sprint. Using this formula, the planning meeting for a two-week sprint would last two hours, although this can vary.		At the end of the sprint, the team presents its completed work to the project owner and the project owner uses the criteria established at the sprint planning meeting to either accept or reject the work.
Story Point*			Agile teams generally prefer to express estimates in units other than the time-honored "man-day" or "man-hour." Possibly the most widespread unit is "story points."	Story points are a non-unit measure used to determine the complexity of a user story. Story points are relative, not absolute, and do not relate to actual hours—they can be anything from tee-shirt sizes to the Fibonacci Sequence.		A story point is a metric used in agile project management and development to determine (or estimate) the difficulty of implementing a given story. In this context, a story is a particular business need assigned to the software development team. Using estimations of story points rather than time allows development teams to be less precise. It may be difficult, for example, to estimate how long a particular feature will take to develop but relatively easy to understand if it is more complex than others, in which case it should be assigned more story points.
Test Driven Development			"Test-driven development" is a style of programming in which three activities are tightly interwoven: coding, testing (in the form of writing unit tests) and design (in the form of refactoring).	Test-Driven Development is the practice of designing and building tests for functional, working code, and then building code that will pass those tests.		Test-driven development (TDD), also called test-driven design, is a method of implementing software programming that interlaces unit testing, programming and refactoring on source code. Before any new code is written, the programmer must first create a failing unit test. Then, the programmer -- or pair, or mob -- creates just enough code to satisfy that requirement. Once the test is passing, the programmer may refactor the design, making improvements without changing the behavior.
User Story*		Stories are short descriptions of pieces of desired functionality, written in the user's language. Agile Teams implement small, vertical slices of system functionality and are sized so the can be completed in a single iteration.	In consultation with the customer or product owner, the team divides up the work to be done into functional increments called "user stories."	A user story is a brief, non-technical description of a software system requirement written from the customer's or end-user's point of view. Either the product owner or the team writes the user stories according to the following structure: as a <type of user>, I want to <perform some task> so I can <achieve some goal.>	a description of functionality or value written from the perspective of the user. Each story should be made available to users as an incremental improvement; however, it often makes sense to collect a set of user stories that will be promoted or validated together as a release.	
Velocity			At the end of each iteration, the team adds up effort estimates associated with user stories that were completed during that iteration. This total is called velocity.	Velocity is a metric that specifies how much work a team is able to complete within a single, fixed-length iteration or sprint.	Any measure of the value produced during each sprint. Average velocity can help guide time-boxed iterations. **Velocity is most useful for planning and tracking purposes and does not measure team productivity or performance.**	Velocity is a metric that predicts how much work an Agile software development team can successfully complete within a two-week sprint (or similar time-boxed period). Velocity is a useful planning tool for estimating how fast work can be completed and how long it will take to complete a project. The metric is calculated by reviewing work the team successfully completed during previous sprints; for example, if the team completed 10 stories during a two-week sprint and each story was worth 3 story points, then the team's velocity is 30 story points per sprint.
User/End User	The actual person operating the system in the field; not a representative system engineer representing the end user.					In information technology, the term end user is used to **distinguish the person for whom a hardware or software product is designed** from the developers, installers, and servicers of the product. The "end" part of the term probably derives from the fact that most information technologies involve a chain of interconnected product components at the end of which is the "user." Frequently, complex products require the involvement of other-than-end users such as installers, administrators, and system operators. The term end user thus distinguishes the user for which the product is designed from other users who are making the product possible for the end user. Often, the term user would suffice.
Waterfall						The waterfall model is a linear, sequential approach to the software development life cycle (SDLC) that is popular in software engineering and product development. The waterfall model emphasizes a logical progression of steps. Similar to the direction water flows over the edge of a cliff, distinct endpoints or goals are set for each phase of development and cannot be revisited after completion. The term was first introduced in a paper published in 1970 by Dr. Winston W. Royce and continues to be used in applications of industrial design.
Waterscrumfall						Water-Scrum-fall is a hybrid approach to application lifecycle management that combines waterfall and Scrum development methodologies. Generally speaking, a development team that uses a waterfall approach regards the development process for a software product as one large project. At the end of the project, the team releases working software to an operations team for installation and maintenance. Typically, the business owner (also called the product owner) only sees the finished product.