**What is Devops and what is its concept?**

DevOps is a term for a group of concepts that, while not all new, have catalyzed into a movement and are rapidly spreading throughout the technical community.

**Why Devops is required in today’s Life?**

DevOps describes a culture and set of processes that bring development and operations teams together to complete software development. It allows organizations to create and improve products at a faster pace than they can with traditional software development approaches. And, it’s gaining popularity at a rapid rate.

* Shorter Development Cycles, Faster Innovation
* Reduced Deployment Failures, Rollbacks, and Time to Recover
* Improved Communication and Collaboration
* Increased Efficiencies
* Reduced Costs and IT Headcount

**Explain Devops Life cycle in detail describing each step.**

DevOps defines an agile relationship between development and operations. It is the process practiced by the development team and operational engineers together from beginning to the final stage of the product. Understanding DevOps is not complete without understanding the DevOps lifecycle phases. The continuous DevOps lifecycle includes seven phases as given below.

* Continuous Development
* Continuous Integration
* Continuous Testing
* Continuous Monitoring
* Continuous Feedback
* Continuous Deployment
* Continuous Operations

**Different stages in DevOps:**

* Continuous Development: in waterfall model, the software product parts broken into multiple parts for the short development cycles, but in this stage, your software is getting developed continuously.
* Continuous Integration: In this stage, if something new is announced in the product, then we will again integrate the thing into the product.
* Continuous testing: the software is tested continuously to detect the bugs.
* Continuous Monitoring: We need to monitor the application again and again and update the application for enhancement.

**Benefits of DevOps.**

* It provides increased software quality.
* Allows us to take feedback in shorter feedback loops.
* Fewer bugs and no downtime.
* Immutable infrastructure as code.

**Agile Model:**

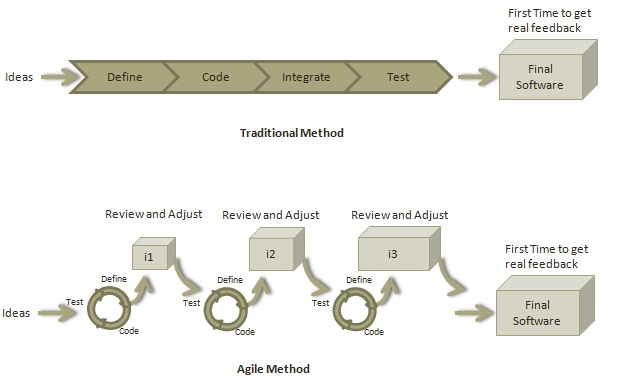
Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations.

**Difference between Agile and Devops:**

|  |  |  |
| --- | --- | --- |
| Parameter | Agile | DevOps |
| What is it? | Agile refers to an iterative approach which focuses on collaboration, customer feedback, and small, rapid releases. | DevOps is considered a practice of bringing development and operations teams together. |
| Purpose | Agile helps to manage complex projects. | DevOps central concept is to manage end-to-end engineering processes. |
| Task | Agile process focusses on constant changes. | DevOps focuses on constant testing and delivery. |
| Implementation | Agile method can be implemented within a range of tactical frameworks like a sprint, safe and scrum. | The primary goal of DevOps is to focus on collaboration, so it doesn't have any commonly accepted framework. |
| Team skill set | Agile development emphasizes training all team members to have a wide variety of similar and equal skills. | DevOps divides and spreads the skill set between the development and operation teams. |
| Team size | Small Team is at the core of Agile. As smaller is the team, the fewer people on it, the faster they can move. | Relatively larger team size as it involves all the stack holders. |
| Duration | Agile development is managed in units of "sprints." This time is much less than a month for each sprint. | DevOps strives for deadlines and benchmarks with major releases. The ideal goal is to deliver code to production DAILY or every few hours. |
| Feedback | Feedback is given by the customer. | Feedback comes from the internal team. |
| Target Areas | Software Development | End-to-end business solution and fast delivery. |
| Shift-Left Principles | Leverage shift-left | Leverage both shifts left and right. |
| Emphasis | Agile emphasizes on software development methodology for developing software. When the software is developed and released, the agile team will not care what happens to it. | DevOps is all about taking software which is ready for release and deploying it in a reliable and secure manner. |
| Cross-functional | Any team member should be able to do what's required for the progress of the project. Also, when each team member can perform every job, it increases understanding and bonding between them. | In DevOps, development teams and operational teams are separate. So, communication is quite complex. |

**Agile**

Agile is a software development methodology to build a software incrementally using short iterations of 1 to 4 weeks so that the development process is aligned with the changing business needs. Instead of a single-pass development of 6 to 18 months where all the requirements and risks are predicted upfront, Agile adopts a process of frequent feedback where a workable product is delivered after 1 to 4 week iteration.



Roles in Agile

Scrum Master

A Scrum Master is a team leader and facilitator who helps the team members to follow agile practices so that they can meet their commitments. The responsibilities of a scrum master are as follows −

* To enable close co-operation between all roles and functions.
* To remove any blocks.
* To shield the team from any disturbances.
* To work with the organization to track the progress and processes of the company.
* To ensure that Agile Inspect & Adapt processes are leveraged properly which includes
  + Daily stand-ups,
  + Planned meetings,
  + Demo,
  + Review,
  + Retrospective Meetings, and
  + To facilitate team meetings and decision-making process.

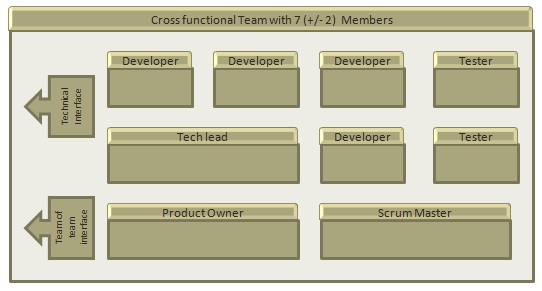
Product Owner

A Product Owner is the one who drives the product from business perspective. The responsibilities or a Product Owner are as follows −

* To define the requirements and prioritize their values.
* To determine the release date and contents.
* To take an active role in iteration planning and release planning meetings.
* To ensure that team is working on the most valued requirement.
* To represent the voice of the customer.
* To accept the user stories that meet the definition of done and defined acceptance criteria.

Cross-functional Team

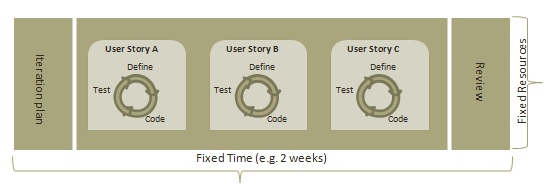
Every agile team should be a self-sufficient team with 5 to 9 team members and an average experience ranging from of 6 to 10 years. Typically, an agile team comprises of 3 to 4 developers, 1 tester, 1 technical lead, 1 product owner and 1 scrum master.



Product Owner and Scrum master are considered to be a part of Team Interface, whereas other members are part of Technical Interface.

How an Agile Team Plans its Work?

An Agile team works in iterations to deliver user stories where each iteration is of 10 to 15 days. Each user story is planned based on its backlog prioritization and size. The team uses its capacity − how many hours are available with team to work on tasks − to decide how much scope they have to plan.



Point

A Point defines how much a team can commit. A point usually refers to 8 hours. Each story is estimated in points.

Capacity

Capacity defines how much an individual can commit. Capacity is estimated in hours.

What is a User Story?

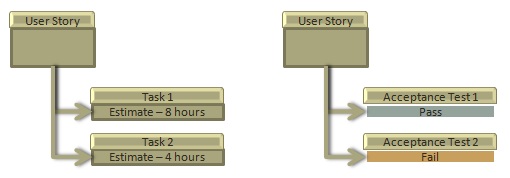
A user story is a requirement which defines what is required by the user as functionality. A user story can be in two forms −

* As a <User Role> I want <Functionality> so that <Business Value>
* In order to <Business value> as a <User Role> I want <Functionality>

During release planning, a rough estimate is given to a user story using relative scale as points. During iteration planning, the story is broken down into tasks.

Relationship of User Stories and Tasks

* User story talks about what is to be done. It defines what a user needs.
* Task talks about how it is to be done. It defines how a functionality is to be implemented.
* Stories are implemented by tasks. Each story is a collection of tasks.
* User story is divided into tasks when it is planned in current iteration.
* Tasks are estimated in hours, typically from 2 to 12 hours.
* Stories are validated using acceptance tests.



When a Story is Done

The team decides what **done** means. The criteria may be −

* All tasks (development, testing) are completed.
* All acceptance tests are running and are passed.
* No defect is open.
* Product owner has accepted the story.
* Deliverable to the end-user.

What is Acceptance Criteria?

Criteria defines the functionality, behavior, and performance required by a feature so that it can be accepted by the product owner. It defines what is to be done so that the developer knows when a user story is complete.

How the Requirements are Defined?

Requirements are defined as

* A User Story,
* With Acceptance Criteria, and
* Tasks to implement the story.

# **Agile – Manifesto**

## Twelve Principles of Agile Manifesto

* **Customer Satisfaction** − Highest priority is given to satisfy the requirements of customers through early and continuous delivery of valuable software.
* **Welcome Change** − Changes are inevitable during software development. Ever-changing requirements should be welcome, even late in the development phase. Agile processes should work to increase customers' competitive advantage.
* **Deliver a Working Software** − Deliver a working software frequently, ranging from a few weeks to a few months, considering shorter time-scale.
* **Collaboration** − Business people and developers must work together during the entire life of a project.
* **Motivation** − Projects should be built around motivated individuals. Provide an environment to support individual team members and trust them so as to make them feel responsible to get the job done.
* **Face-to-face Conversation** − Face-to-face conversation is the most efficient and effective method of conveying information to and within a development team.
* **Measure the Progress as per the Working Software** − Working software is the key and it should be the primary measure of progress.
* **Maintain Constant Pace** − Agile processes aim towards sustainable development. The business, the developers, and the users should be able to maintain a constant pace with the project.
* **Monitoring** − Pay regular attention to technical excellence and good design to enhance agility.
* **Simplicity** − Keep things simple and use simple terms to measure the work that is not completed.
* **Self-organized Teams** − An agile team should be self-organized and should not depend heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams.
* **Review the Work Regularly** − Review the work done at regular intervals so that the team can reflect on how to become more effective and adjust its behavior accordingly.

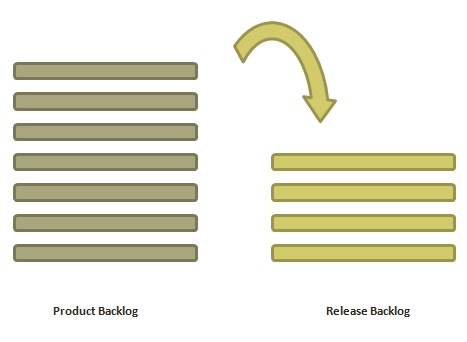
# **Agile - Characteristics**

## Iterative/incremental and Ready to Evolve

Most of the agile development methods break a problem into smaller tasks. There is no direct long-term planning for any requirement. Normally, iterations are planned which are of vary short period of time, for example, 1 to 4 weeks. A cross-functional team is created for each iteration that works in all functions of software development like planning, requirements analysis, design, coding, unit testing, and acceptance testing. The result at the end of the iteration is a working product and it is demonstrated to the stakeholders at the end of an iteration.

# **Agile - Release Planning**

The purpose of release planning is to create a plan to deliver an increment to the product. It is done after every 2 to 3 months.



Who is Involved?

* **Scrum Master** − The scrum master acts as a facilitator for the agile delivery team.
* **Product Owner** − The product owner represents the general view of the product backlog.
* **Agile Team** − Agile delivery team provides insights on the technical feasibilities or any dependencies.
* **Stakeholders** − Stakeholders like customers, program managers, subject matter experts act as advisers as decisions are made around the release planning.

Prerequisites of Planning

The prerequisites of release planning are as follows −

* A ranked product backlog, managed by the Product Owner. Generally five to ten features are taken which the product owner feels that can be included in a release
* Team's input about capabilities, known velocity or about any technical challenge
* High-level vision
* Market and Business objective
* Acknowledgement whether new product backlog items are needed

Materials Required

The list of materials required for release planning is as follows −

* Posted agenda, purpose
* Flip charts, whiteboards, markers
* Projector, way to share computers having data/tools required during planning meeting
* Planning data

Planning Data

The list of data required to do release planning is as follows −

* Previous iterations or release planning results
* Feedback from various stakeholders on product, market conditions, and deadlines
* Action plans of previous releases / iterations
* Features or defects to be considered
* Velocity from previous releases/ estimates.
* Organizational and personal calendars
* Inputs from other teams and subject matter experts to manage any dependencies

Output

The output of a release planning can be the following −

* Release plan
* Commitment
* Issues, concerns, dependencies, and assumptions which are to be monitored
* Suggestions to improve future release plannings

Agenda

The agenda of a release planning can be −

* **Opening ceremony** − Welcome message, review purpose and agenda, organizing tools and introduction to business sponsors.
* **Product Vision, Roadmap** − Show the large picture of the product.
* **Review previous releases** − Discussion on any item which can impact the plan.
* **Release name / theme** − Inspect the current status of roadmap themes and do the required adjustments, if any.
* **Velocity** − Present the velocity for the current release and of previous releases.
* **Release schedule** − Review key milestones and decision on time boxes for release and iterations within release.
* **Issues and concerns** − Check any concerns or issue and record them.
* **Review and Update the Definition of Done** − Review the definition of **done** and make appropriate changes based on technology, skill, or changes in team members since the last iteration / release.
* **Stories and items to be considered** − Present the user stories and features from the product backlog to be considered for scheduling in the current release.
* **Determine sizing values** − If the velocity is unknown, then plan the sizing values to be used in the release planning.
* **Coarse the size of stories** − The delivery team determines the appropriate size of the stories under consideration and splits the stories into multiple iterations if a story is too large. The product owner and the subject matter experts clarify the doubts, elaborate the acceptance criteria, and make proper story splits. The scrum master facilitates the collaboration.
* **Map stories to iterations** − The delivery team and the product owner move the stories/defects in the iterations based on the size and velocity. The scrum master facilitates the collaboration.
* **New concerns or issues** − Check any new issues based on previous experience and record the same.
* **Dependencies and assumptions** − Check any dependencies/assumptions planned during the release planning.
* **Commit** − The scrum master calls for the planning. Delivery team and Product owner signal it as the best plan and then commit to move to the next level of planning, that is, iteration planning.
* **Communication and logistics planning** − Review/Update the communication and logistics planning for the release.
* **Parking lot** − Process parking lot means all items should be either resolved or set as action items.
* **Distribute Action items and action plans** − Distribute the action items among their owners, process the action plan.
* **Retrospect** − Solicit feedback from participants to make the meeting successful.
* **Close** − Celebrate the success.

# **Agile - Product Backlog**

A product backlog is a list of items to be done. Items are ranked with feature descriptions. In an ideal scenario, items should be broken down into user stories.

Why Product Backlog is Important?

* It is prepared so that estimates can be given to each and every feature.
* It helps in planning the roadmap for the product.
* It helps in re-ranking the features so that more value can be added to the product.
* It helps in determining what to prioritize first. Team ranks the item and then builds value.

Characteristics of Product Backlog

* Each product should have one product backlog which can have a set of large to very large features.
* Multiple teams can work on a single product backlog.
* Ranking of features is done based on business value, technical value, risk management or strategic fitness.
* Highest ranking items are decomposed into smaller stories during release planning so that they can be completed in future iterations.