innovate achieve lead

BITS Pilani
Pilani Campus

SE ZG501
Software Quality Assurance and Testing
Session 14

- In Agile, testing is an ongoing activity rather than a distinct phase, focusing on collaboration between testers, developers, and stakeholders.
- This approach helps detect and fix issues early, aligning with Agile's iterative development cycle.

Agile Testing Principles

Shortening feedback iteration: In Agile Testing, the testing team gets to know the product development and its quality for each and every iteration. Thus continuous feedback minimizes the feedback response time and the fixing cost is also reduced.

Testing is performed alongside Agile testing is not a different phase. It is performed alongside the development phase. It ensures that the features implemented during that iteration are actually done. Testing is not kept pending for a later phase.

Involvement of all members: Agile testing involves each and every member of the development team and the testing team. It includes various developers and experts.

Documentation is weightless: In Instead of creating large, detailed documents, agile testers rely on simple and reusable checklists to plan tests. They focus on the core purpose of testing, avoiding unnecessary details, and use lightweight documentation tools to stay efficient

Clean code: The defects that are detected are fixed within the same iteration. This ensures clean code at any stage of development.

Constant response: Agile testing helps to deliver responses or feedback on an ongoing basis. Thus, the product can meet the business needs.

Customer satisfaction: In agile testing, customers are exposed to the product throughout the development process. Throughout the development process, the customer can modify the requirements, and update the requirements and the tests can also be changed as per the changed requirements.

Test-driven: In agile testing, testing is performed **together with development** to speed up the overall process. In contrast, in traditional methods, testing usually starts **only after** the software has been fully developed.

Agile Testing Methodologies

Test-Driven Development (TDD): TDD is the software development process relying on creating unit test cases before developing the actual code of the software. It is an iterative approach that combines 3 operations, programming, creation of unit tests, and refactoring.

Behavior Driven Development (BDD): BDD is agile software testing that aims to document and develop the application around the user behavior a user expects to experience when interacting with the application. It encourages collaboration among the developer, quality experts, and customer representatives.

Exploratory Testing: In exploratory testing, the tester has the freedom to explore the code and create effective and efficient software. It helps to discover the unknown risks and explore each aspect of the software functionality.

Acceptance Test-Driven Development (ATDD): ATDD is a collaborative process where customer representatives, developers, and testers come together to discuss the **requirements, and potential pitfalls** and thus **reduce the chance of errors before coding begins.**

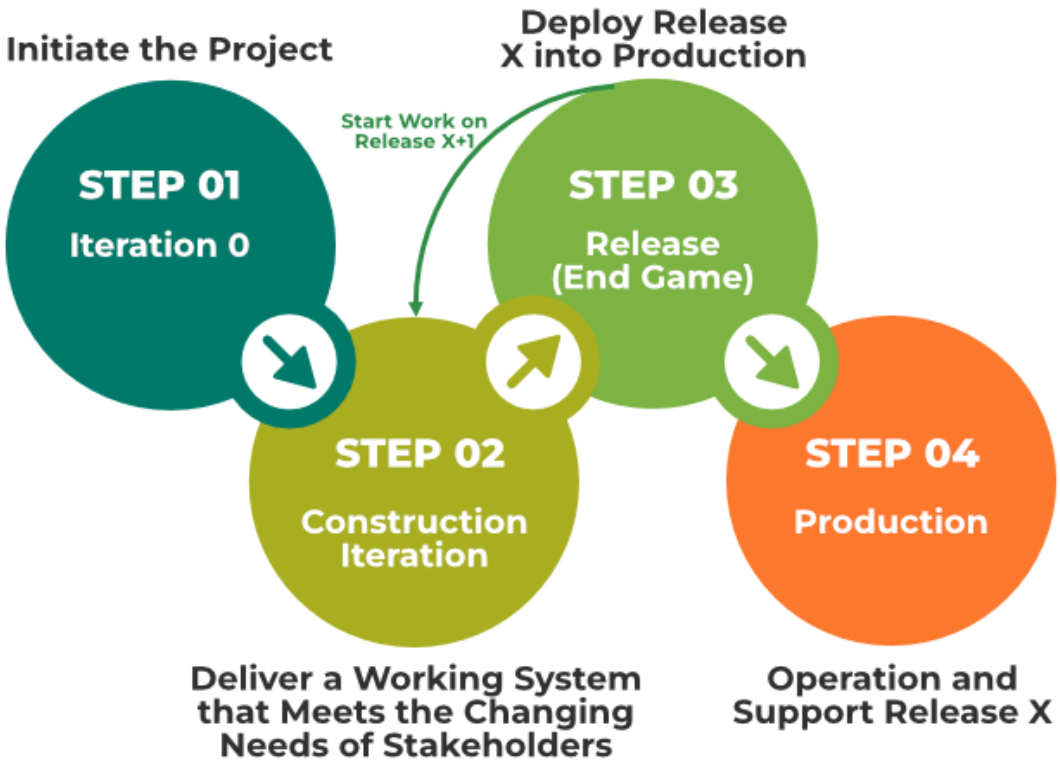
Extreme Programming (XP): Extreme programming is a customer-oriented methodology that helps to deliver a good quality product that meets customer expectations and requirements.

Session-Based Testing: It is a structured and time-based approach that involves the progress of exploratory testing in multiple sessions. This involves uninterrupted testing sessions that are time-boxed with a duration varying from 45 to 90 minutes. During the session, the tester creates a document called a **charter document** that includes various information about their testing.

Dynamic Software Development Method (DSDM): DSDM is an agile project delivery framework that provides a framework for building and maintaining systems. It can be used by users, developers, and testers.

Crystal Methodologies: This methodology focuses on people and their interactions when working on the project instead of processes and tools. The suitability of the crystal method depends on three dimensions, team size, criticality, and priority of the project.

Agile Testing Strategies



Iteration 0

Iteration 0:

- This is the **first stage** of the testing process where the **initial setup** is done.
- **The testing environment is created** in this phase.

Key activities:

- Find people for testing, set up the usability testing lab, and prepare necessary resources.
- Verify the project's business case, project boundaries, and scope.
- Summarize important requirements and use cases.
- Plan the project's initial costs and valuation.
- Identify potential risks.
- Create one or more possible design options for the project.

Construction Iteration

It is the second phase of the testing process and the main phase where most of the work happens.

It includes multiple smaller iterations to gradually build and improve the solution.

Confirmatory testing: This type of testing concentrates on Focuses on **checking if the system meets stakeholder requirements** as shared with the team so far. It is performed by the development team and has two types:

Agile acceptance testing: It is the combination of acceptance testing and functional testing. It can be executed by the development team and the stakeholders.

Developer testing: It is the combination of unit testing and integration testing and verifies both the application code and database schema.

Investigative Testing:

- This testing **finds problems that were missed** during confirmatory testing.
- Testers look for **hidden or unexpected issues** by creating **defect stories**.

Focus areas:

- Integration testing
- Load testing
- Security testing
- Stress testing

Release End Game

Release End Game (Transition Phase):

- This is the **final phase** before product release.
- It includes **full system testing** and **acceptance testing**.
- The product is **tested very thoroughly** to find and fix any remaining issues (defect stories).

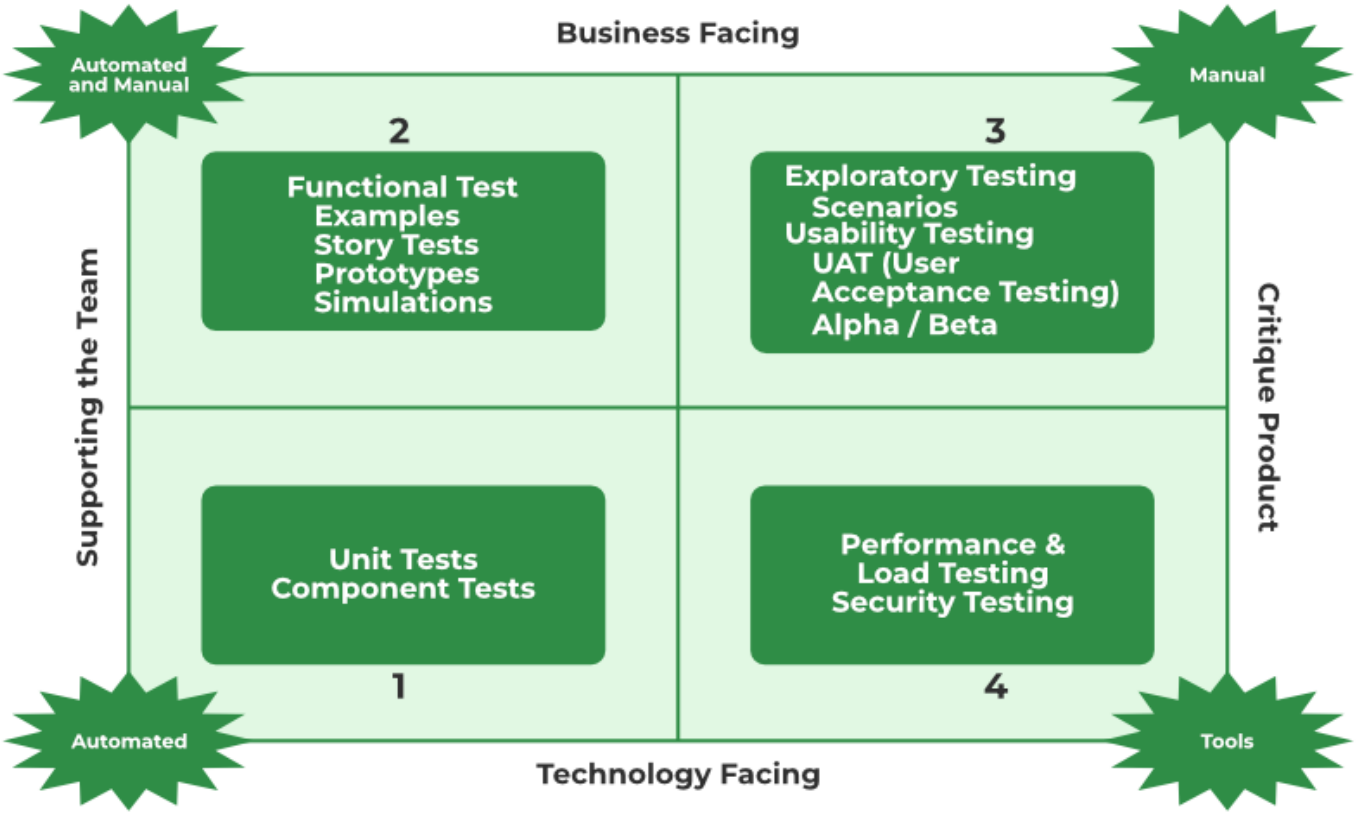
Key activities in this phase:

- Train the end-users.
- Support operational and service people.
- Plan and execute product marketing.
- Set up backup and recovery systems.
- Finalize system and user documentation.

Production

It is the last phase of agile testing. The product is finalized in this stage after the removal of all defects and issues raised.

Agile Testing Quadrants



Quadrant 1 (Automated)

The first agile quadrat focuses on the internal quality of code which contains the test cases and test components that are executed by the test engineers.

All test cases are technology-driven and used for automation testing. All through the agile first quadrant of testing, the following testing can be executed:

- Unit testing.
- Component testing.

Quadrant 2 (Manual and Automated)

The second agile quadrant focuses on the customer requirements that are provided to the testing team before and throughout the testing process. The test cases in this quadrant are **business-driven** and are used for manual and automated functional testing. The following testing will be executed in this quadrant:

- Pair testing.
- Testing scenarios and workflow.
- Testing user stories and experiences like prototypes.

Quadrant 3 (Manual)

Quadrant 3 (Manual Testing):

- This quadrant gives **feedback** to Quadrant 1 (internal quality) and Quadrant 2 (business needs).
- It involves **multiple rounds of manual testing** to **review, strengthen, and improve the code.**
- The feedback collected here is used to **prepare better automation tests** later.

Main types of testing done:

- **Usability Testing:** Check how easy and friendly the system is for users.
- **Collaborative Testing:** Testers and others (like developers or customers) work together to find issues.
- **User Acceptance Testing (UAT):** Verify if the system meets customer requirements.
- **Pair Testing with Customers:** Testers and customers test together to explore the system.

Quadrant 4 (Tools)

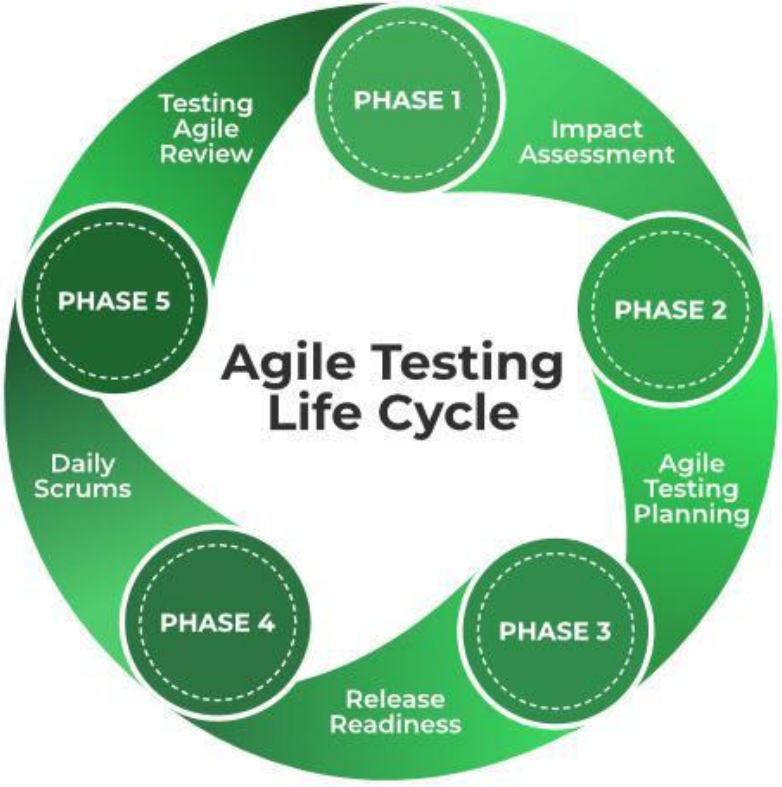
Quadrant 4 (Tools):

- Focuses on the **non-functional requirements** of the product like **performance**, **security**, and **stability**.
- The goal is to ensure the system is **strong, secure, and scalable**.
- This quadrant uses **special testing tools** to measure system qualities and expected performance.

Main types of testing done:

- **Non-functional Testing:** Stress testing, load testing, performance testing.
- **Security Testing:** Check for vulnerabilities and risks.
- **Scalability Testing:** Ensure the system can handle growth (more users, data).
- **Infrastructure Testing:** Check server, network, cloud setup.
- **Data Migration Testing:** Ensure data moves safely between systems.

Agile Testing Life Cycle



Agile Testing Life Cycle :
5 different phases:

Impact Assessment: This is the first phase of the agile testing life cycle also known as the feedback phase where the inputs and responses are collected from the users and stakeholders. This phase supports the test engineers to set the objective for the next phase in the cycle.

Agile Testing Planning: In this phase, the developers, customers, test engineers, and stakeholders team up to plan the testing process schedules, regular meetings, and deliverables.

Release Readiness: This is the third phase in the agile testing lifecycle where the test engineers review the features which have been created entirely and test if the features are ready to go live or not and the features that need to be sent again to the previous development phase.

Daily Scrums: This phase involves the daily morning meetings to check on testing and determine the objectives for the day. The goals are set daily to enable test engineers to understand the status of testing.

Test Agility Review: This is the last phase of the agile testing lifecycle that includes weekly meetings with the stakeholders to evaluate and assess the progress against the goals.

Brain Teaser Break

During an Agile sprint, a team is preparing to validate new features before release. The team involves a tester and developer working side-by-side, discusses how a user will complete a purchase flow, and simulates this flow using early interface designs.

Which of the following testing practices from **Quadrant 2** are being applied in this scenario?
(Select all that apply)

A) Pair Testing
B) Exploratory Testing
C) Workflow and Scenario Testing
D) Prototype-based Testing

Correct Answers: A, C, D

Agile Test Plan

innovate achieve lead

1. **Purpose**:
    1. In Agile development, a test plan is essential for every iteration or release.
2. **What It Includes**:
    1. **Types of Testing**: Specifies the kinds of tests (e.g., unit, integration, regression) that will be conducted.
    2. **Test Data Requirements**: Defines what test data is needed.
    3. **Test Environments**: Describes where the tests will be executed (e.g., staging servers, test labs).
    4. **Test Results**: Captures and evaluates the outcomes of the testing activities.
3. **Frequency**:
    1. A new **test plan is created and updated for every release** to reflect changes in features or code.

The Agile test plan includes the following:

- Test Scope.
- Testing instruments.
- Data and settings are to be used for the test.
- Approaches and strategies used to test.
- Skills required to test.
- New functionalities are being tested.
- Levels or Types of testing based on the complexity of the features.
- Resourcing.
- Deliverables and Milestones.
- Infrastructure Consideration.
- Load or Performance Testing.
- Mitigation or Risks Plan.

Benefits of Agile Testing

Saves time: Implementing agile testing helps to make cost estimates more transparent and thus helps to save time and money.

Reduces documentation: It requires less documentation to execute agile testing.

Enhances software productivity: Agile testing helps to reduce errors, improve product quality, and enhance software productivity.

Higher efficiency: In agile software testing the work is divided into small parts thus developer can focus more easily and complete one part first and then move on to the next part. This approach helps to identify minor inconsistencies and higher efficiency.

Improve product quality: In agile testing, regular feedback is obtained from the user and other stakeholders, which helps to enhance the software product quality.

Limitations of Agile Testing

Project failure: In agile testing, if one or more members leave the job then there are chances for the project failure.

Limited documentation: In agile testing, there is no or less documentation which makes it difficult to predict the expected results as there are explicit conditions and requirements.

Introduce new bugs: In agile software testing, bug fixes, modifications, and releases happen repeatedly which may sometimes result in the introduction of new bugs in the system.

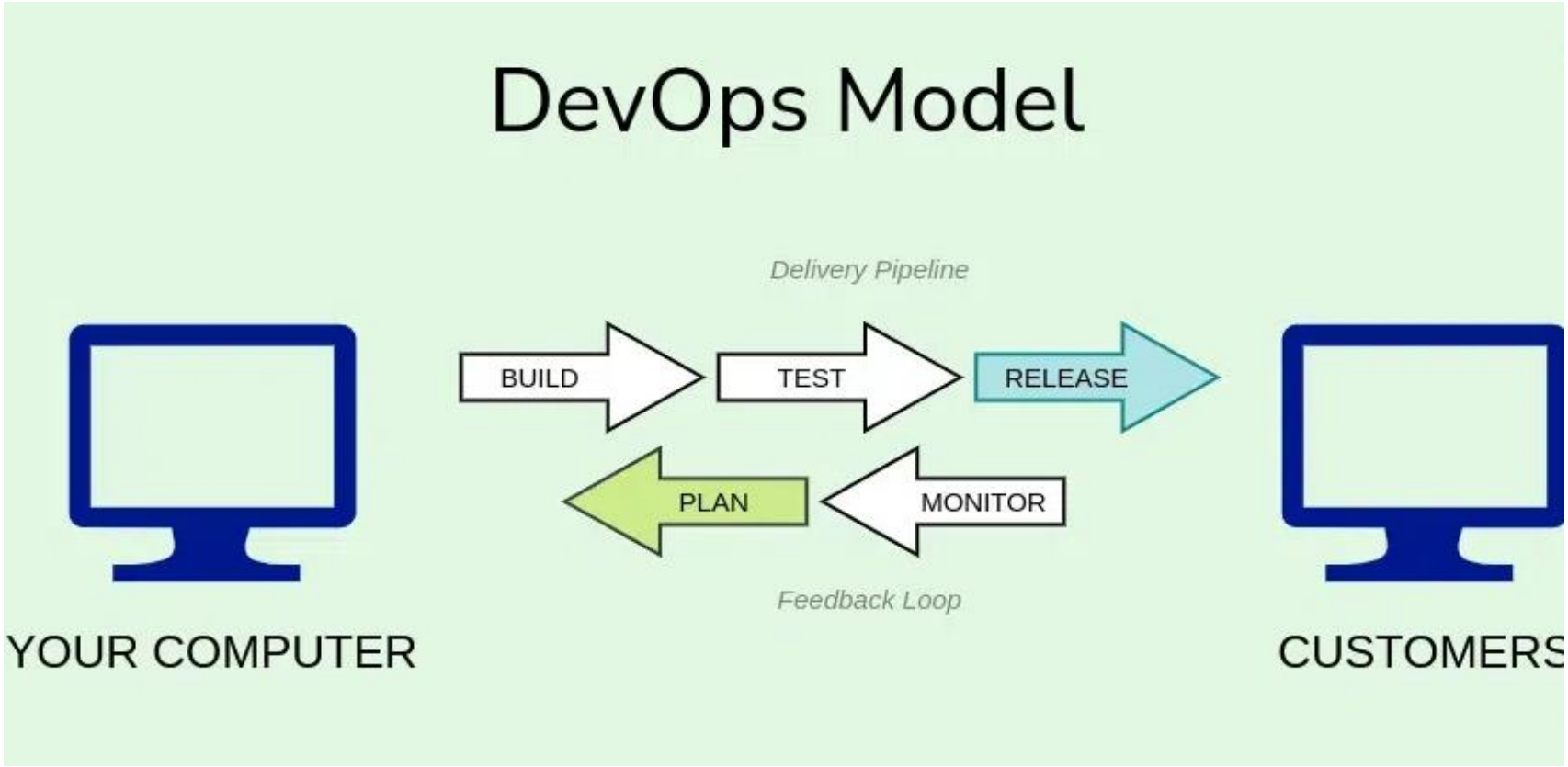
Poor planning: In agile testing, the team is not exactly aware of the end result from day one, so it becomes challenging to predict factors like cost, time, and resources required at the beginning of the project.

No finite end: Agile testing requires minimal planning at the beginning so it becomes easy to get sidetracked while delivering the new product. There is no finite end and there is no clear vision of what the final product will look like.

DEVOPS

- As technology grows and changes fast, there is a **huge need** for **faster and more reliable software delivery**.
- **DevOps** is introduced to **connect software development and IT operations**, making the process **smoother, faster, and more efficient**.

- DevOps is a **culture and practice** that **brings software development (Dev) and IT operations (Ops) together**.
- It encourages **teamwork** and **uses automation tools** to make software releases **faster, smoother, and more reliable**.
- DevOps focuses on **strong communication and collaboration** between developers and operations teams.
- The goal is to **speed up software delivery** and **improve the quality and reliability** of the final product.


DevOps Model
Delivery Pipeline
BUILD
TEST
RELEASE
PLAN
MONITOR
Feedback Loop
YOUR COMPUTER
CUSTOMERS

Delivery Pipeline

The pipeline represents the different stages that software goes through before it is released to production. These stages might typically include:

Build: The stage where the software code is compiled and packaged into a deployable unit.

Test: The stage where the software is rigorously tested to ensure it functions as expected and identifies any bugs.

Release: The stage where the software is deployed to production for end users.

Feedback Loop

The loop indicates that information and learnings from the production environment are fed back into the earlier stages of the pipeline. This feedback can be used to improve the software development process and future releases.

How DevOps Works?

- DevOps will remove the Isolation conditions between the development team and operations team.
- In many cases these two teams will work together for the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.

- Teams in charge of security and quality assurance may also integrate more closely with development and operations over the course of an application's lifecycle under various DevOps models.
- DevSecOps is the term used when security is a top priority for all members of a DevOps team.

DevOps Life Cycle

- DevOps is a practice that enables a single team to handle the whole application lifecycle, including development, testing, release, deployment, operation, display, and planning. It is a mix of the terms “Dev” (for development) and “Ops” (for operations).
- We can speed up the delivery of applications and services by a business with the aid of DevOps.
- Amazon, Netflix, and other businesses have all effectively embraced DevOps to improve their customer experience.

- **DevOps Lifecycle** is a series of steps where development and operations teams work together to deliver software faster.
- DevOps uses important steps like:
  coding, building, testing, releasing, deploying, operating, monitoring (displaying), and planning.
- DevOps lifecycle focuses on doing these activities continuously, such as:
  continuous development, integration, testing, monitoring, and feedback.


DevOps Lifecycle
Continuous Development
Continuous Integration
Continuous Testing
Continuous Deployment
Continuous Monitoring
Continuous Feedback
Continuous Operations

7 Cs of DevOps

- Continuous Development
- Continuous Integration
- Continuous Testing
- Continuous Deployment/Continuous Delivery
- Continuous Monitoring
- Continuous Feedback
- Continuous Operations

Benefits of DevOps

Faster Delivery: DevOps enables organizations to release new products and updates faster and more frequently, which can lead to a competitive advantage.

Improved Collaboration: DevOps promotes collaboration between development and operations teams, resulting in better communication, increased efficiency, and reduced friction.

Improved Quality: DevOps emphasizes automated testing and continuous integration, which helps to catch bugs early in the development process and improve the overall quality of software.

Increased Automation: DevOps enables organizations to automate many manual processes, freeing up time for more strategic work and reducing the risk of human error.

Better Scalability: DevOps enables organizations to quickly and efficiently scale their infrastructure to meet changing demands, improving the ability to respond to business needs.

Increased Customer Satisfaction: DevOps helps organizations to deliver new features and updates more quickly, which can result in increased customer satisfaction and loyalty.

Improved Security: DevOps promotes security best practices, such as **continuous testing and monitoring**, which can help to reduce the risk of security breaches and improve the overall security of an organization's systems.

Better Resource Utilization: DevOps enables organizations to optimize their use of resources, including hardware, software, and personnel, which can result in cost savings and improved efficiency

Continuous Testing in DevOps

- Fixing a software bug after a product is launched can be up to 100 times more expensive than addressing it during the development phase.
- Continuous Testing is a vital approach in modern software development, where testing is integrated at every stage of the process.
- This helps identify errors early, reducing both the risk of defects and the costs that come with fixing them later.

- **Continuous Testing** refers to running automated tests every time code changes are made, providing fast feedback as part of the software delivery pipeline.
- It was introduced to help developers quickly identify, notify, and fix issues. The goal is to test more frequently, starting with individual components and later testing the entire codebase.
- Continuous Testing is a key part of the Continuous Integration (CI) process within Agile and DevOps pipelines, enabling faster and more efficient software delivery.

How is Continuous Testing different?

In Traditional (Hands-off) Testing:

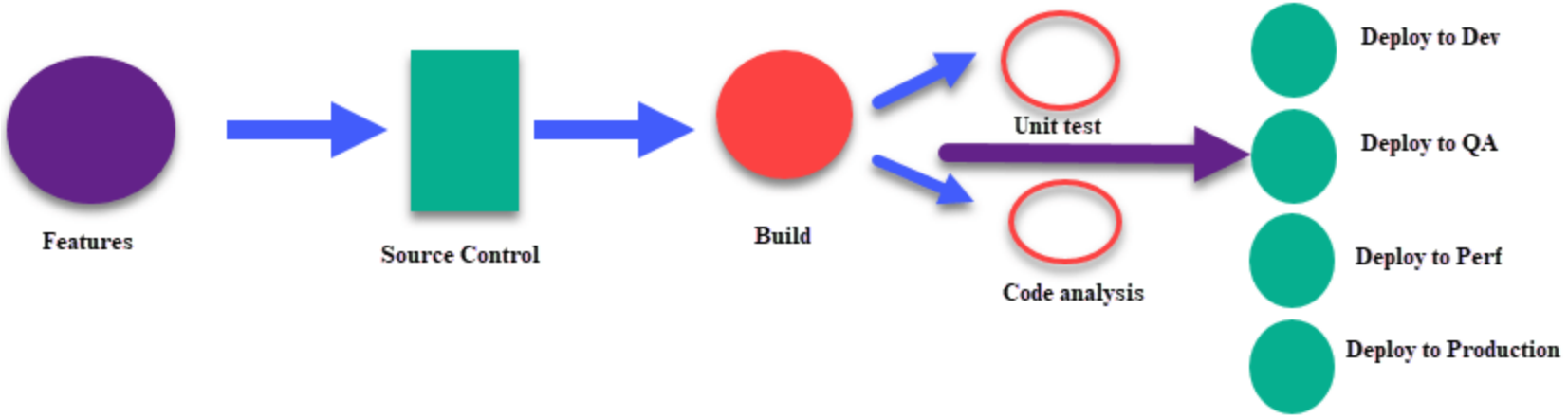
The product owner defines software requirement and stories. hands them off to Development Team

The Development codes and does unit test. Hands off to QA Team

QA Team checks the software and hands off the verified software to Deployment or Developers

- Continuous means undisrupted testing done on a continuous basis.
- In a Continuous DevOps process, a software change (release candidate) is continuously moving from Development to Testing to Deployment.
- The code is continuously developed, delivered, tested and deployed.

Continuous Testing



For Example,

- Whenever a developer uploads the code in the Source Code Server like Jenkins automated set of unit tests are executed in the continuous process.
- If the tests fail, the build is rejected, and the developer is notified. If the build passes the test, it is deployed to performance, QA servers for exhaustive functional and load tests.
- The tests are run in parallel. If the tests pass, the software is deployed in production.