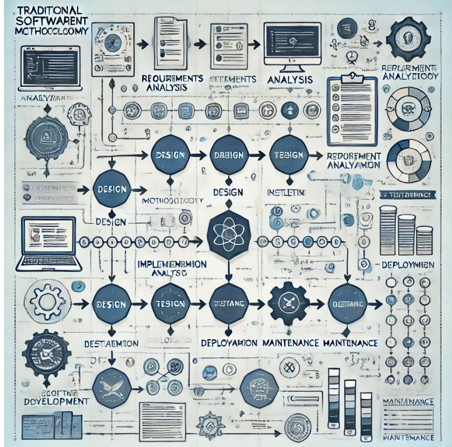
DEVOPS OVERVIEW

Nguyễn Thanh Quân - ntquan@fit.hcmus.edu.vn

Agenda

1. Traditional Software Development Methodology
2. Introduction to DevOps
3. Addressing Challenges through DevOps
4. Agile Methodology
5. The Relationship between Agile and DevOps
6. DevOps Toolchain
7. DevOps Principles
8. DevOps Best Practices

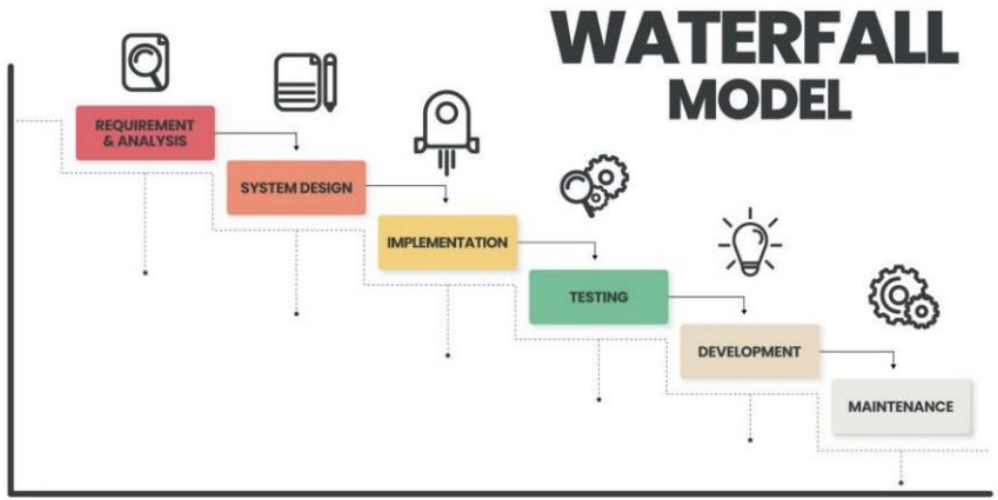


Traditional Software Development Methodology

Traditional Software Development Methodology

Waterfall Methodology:

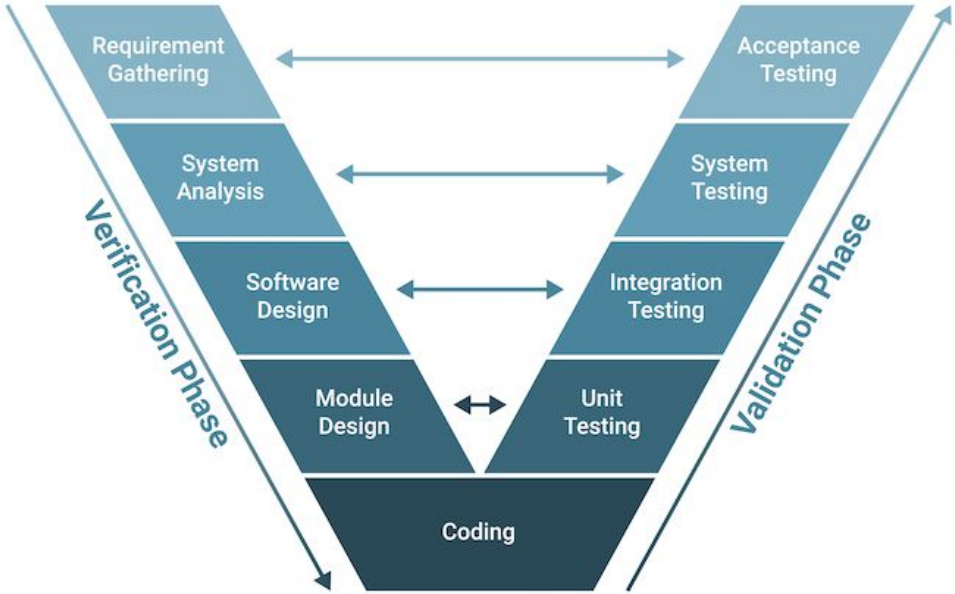
The Waterfall model is a linear sequential software development methodology in which development stages (planning, requirements analysis, design, implementation, testing) are carried out in distinct steps, and there is no way to return to previous steps once a stage is completed.



Traditional Software Development Methodology

V-Model Methodology:

The V-Model is a software development methodology similar to the Waterfall model but integrates development and testing phases. It requires testing to be planned and executed from the very beginning of the software development process.



Traditional Software Development Methodology

Challenges with Traditional Approaches

1. Time and Schedule Challenges

Lengthy and Unpredictable Timeframes: Traditional methodologies often involve long project durations with rigid phase transitions, making it difficult to adapt to changing requirements or new technologies.

Risk of Missing Requirements: Overly detailed planning at the beginning, without early feedback, can result in developing the wrong solutions or failing to meet user needs effectively.

Traditional Software Development Methodology

Challenges with Traditional Approaches

2. Challenges in Managing Changes

Difficulty in Changing Requirements: The Waterfall methodology makes it difficult to accommodate changes during the development process, leading to costly and complex adjustments in later stages of the project.

Errors Arising from Changes: very requirement change affects parts of the system that have already been developed, increasing the likelihood of introducing errors into the software.

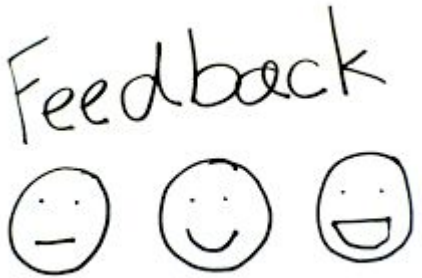
Traditional Software Development Methodology

Challenges with Traditional Approaches

3. Lack of Quick Feedback

Delayed Feedback During Development: Traditional software development teams typically receive feedback from customers only after the software is fully deployed. This can result in the final product failing to meet user needs effectively.

Absence of Continuous Collaboration: This approach does not encourage ongoing collaboration between development teams and customers, making it difficult to implement changes or optimize the product during the development process.



Traditional Software Development Methodology

Challenges with Traditional Approaches

4. Challenges in Testing and Deployment

Late Testing: Testing is typically performed at the end of the process (after the code has been fully developed), which can lead to late detection of errors, increasing the cost and effort required to fix them.

Difficult Deployment: Deploying software into a real-world environment in traditional methodologies is often challenging, as there is no test environment that mirrors the production environment, increasing deployment risks.

Traditional Software Development Methodology

Challenges with Traditional Approaches

5. Waste and Inefficiency

Resource Wastage: Stages like requirements analysis or design may need to be redone multiple times, resulting in wasted time and resources.

Difficulty in Achieving Flexibility: The linear nature of traditional methodologies requires teams to redo previous steps before moving forward, leading to time delays, especially when requirements change.

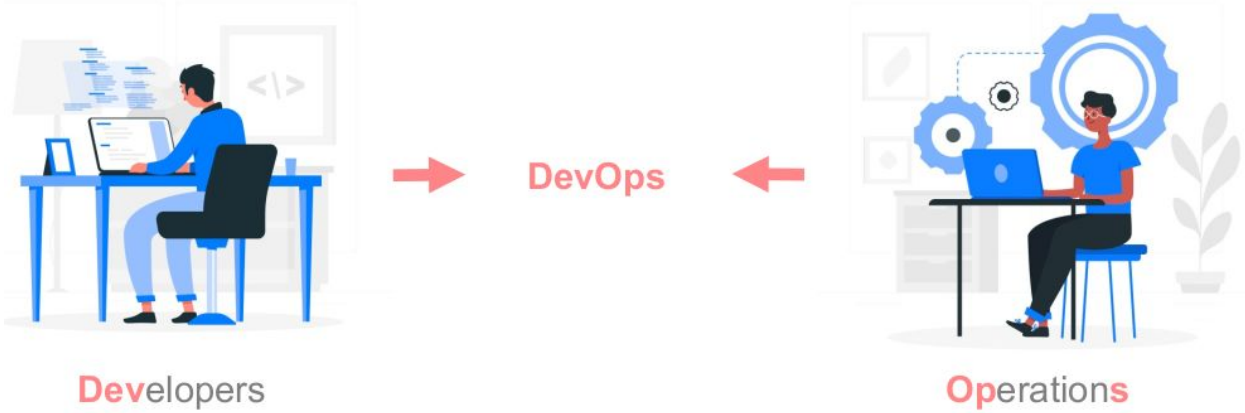
Introduction to DevOps

What is Devops

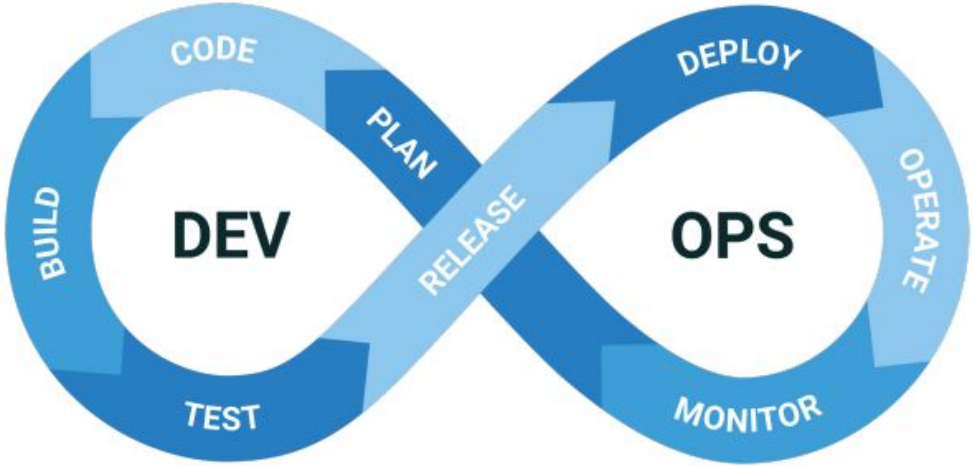
- DevOps is a working methodology applied in the IT software development industry.

- DevOps combines practices, a collaborative working culture, and tools that enhance an organization's ability to deliver applications and services at a high level.

- The name DevOps comes from Developer and Operations. DevOps bridges the communication gap between the software developers and the It operation teams.



Introduction to DevOps



Devops Lifecycle

Introduction to DevOps

1. Plan - This stage involves initial planning for how you envision the development process, including defining project goals and requirements.
2. Code - Writing the code for applications or features according to the organization's or company's requirements.
3. Build - Integrating the written code into a build process, ensuring everything works together properly.
4. Test - Testing the application to ensure that it is ready
5. Releases - If the testing phase is successful, the application is ready to be deployed and made operational.

Introduction to DevOps

6. Deploy - The code is deployed to the production environment.

7. Operate - The application is operated after deployment, ensuring it runs as expected.

8. Monitor - The application is continuously monitored, and necessary changes are made to ensure smooth and stable operation.

Addressing Challenges through DevOps

Addressing Challenges through DevOps

1. Enhancing Development and Deployment Speed

Challenge: Slow software development, unable to quickly respond to changing requirements.

Solution:

- Continuous Integration (CI) and Continuous Deployment (CD): CI/CD automates the process of integrating source code and deploying to testing or production environments, reducing the time from development completion to product release.
- Automated Testing: Automated testing helps quickly identify errors during the development process, speeding up development and ensuring quality.

Addressing Challenges through DevOps

2. Improving Software Quality

Challenge: Slow software development, unable to quickly respond to changing requirements.

Solution:

- Automated Testing: DevOps promotes automating testing to catch errors early, from unit testing to integration testing and user acceptance testing.
- Consistent Environments: DevOps ensures that the development and production environments are similar, minimizing errors during deployment.

Addressing Challenges through DevOps

3. Managing Changes and Deployment Errors

Challenge: Managing changes to source code, new features, or deployment environments is difficult.

Solution:

- Configuration Management: DevOps automates infrastructure and software configuration management using tools like Ansible, Puppet, or Chef.
- Feature Toggles: Feature toggles allow for changes or testing of features without modifying the main source code, enabling rapid deployment without disruption.

Addressing Challenges through DevOps

4. Improving Collaboration Between Dev and Ops Teams

Challenge: Lack of collaboration between development and operations teams leads to deployment and maintenance issues.

Solution:

- Enhancing Collaboration and Communication: DevOps promotes connecting development, testing, and operations teams, helping reduce barriers and improve work efficiency.
- Task Management through Tools: Tools like Slack, JIRA, and Confluence facilitate easier and more effective collaboration, improving communication and progress tracking.

Agile Methodology

AGILE

Agile is a flexible software development methodology that emphasizes collaboration, delivering products early and frequently, while also responding quickly to changes.

AGILE

4 Core Values:

1. Individuals and Interactions over Processes and Tools
2. Working Software over Comprehensive Documentation
3. Collaboration with Customers over Contract Negotiation
4. Responding to Change over Following a Plan

AGILE

Framework Agile

1. Scrum

- Roles: Product Owner, Scrum Master, Development Team.
- Events: Sprint Planning, Daily Standup, Sprint Review, Sprint Retrospective.
- Artifacts: Product Backlog, Sprint Backlog, Increment.

2. Kanban

- Principles: Visualize Work, Limit Work in Progress (WIP), Continuous Improvement
- Tools: Kanban Board

...

AGILE

Benefits of Agile:

- **Reduced Product Development Time**: Agile practices, such as iterative development, allow for faster delivery of features and product releases.
- **Increased Customer Satisfaction**: Continuous feedback from customers and regular product updates ensure that the product meets their needs and expectations.
- **Enhanced Flexibility and Adaptability**: Agile allows teams to easily adapt to changes in requirements, technology, or market conditions, ensuring the product remains relevant and aligned with business goals.

The Relationship between Agile and DevOps

Agile and DevOps

1. Goals: Both Agile and DevOps aim to improve speed, quality, and responsiveness in software development:

- Agile: Focuses on accelerating development through flexible processes, allowing for quick adjustments and regular delivery of value.
- DevOps: Focuses on optimizing deployment and operations of software by automating and continuously integrating processes, ensuring efficient and reliable delivery to production.

Agile and DevOps

2. Agile Prepares the Foundation for DevOps

- Agile Manages the Development Process: Agile creates small, deployable products or increments after each sprint, which aligns well with DevOps, where these small increments are continuously tested and deployed through Continuous Integration (CI) and Continuous Deployment (CD).
- DevOps Deploys Agile Products: After the product is developed using Agile methodologies, DevOps ensures quick and stable deployment through automation and containerization, making the process seamless and reliable.

Agile and DevOps

3. Collaboration in Software Development Stages

Stage	Role of Agile	Role of DevOps
Planning	Flexible planning according to sprints, managing the backlog.	Ensures a consistent environment for deployment.
Development	Develop small increments, frequent communication.	Automates source code testing (CI).
Testing	Continuous feedback from customers and QA teams.	Automates testing and integrates with the pipeline.
Deployment	Not directly involved in deployment.	Automatically deploys source code to production environments.
Operations & Maintenance	Improves software through customer feedback.	Monitors and tracks system performance (monitoring).

Agile and DevOps

4. Culture and Collaboration

Both Agile and DevOps emphasize high levels of collaboration between teams:

- Agile: Focuses on communication and interaction among development team members. It encourages regular collaboration through daily stand-ups, sprint reviews, and feedback loops, fostering teamwork and flexibility.
- DevOps: Bridges the gap between development (Development) and operations (Operations) teams to create a seamless working culture. This integration fosters collaboration, continuous feedback, and efficient processes for deploying and maintaining software in production.

Agile and DevOps

5. Continuous Improvement Capability

Both Agile and DevOps emphasize high levels of collaboration between teams:

- Agile: Improves the product through Sprint Reviews and Retrospectives.
- DevOps: Improves system performance through measurement, monitoring, and automation.