Sanjivani University
School of Engineering and Technology
Department of Artificial Intelligence and DataScience

Application Development Practices (24UETES101)

By,
Souhardya Gayen
Aparupa Chakraborty

CIA - I

Component	Capstone	Details of FA	Bloom's Level	Marks	Rationale
(FA)-1	Introduction to Software Development	Write a brief essay on various software development models	Understand	5	To check students' understanding of fundamental software development concepts.
(FA)-2	Quiz	Online quiz on Agile Methodologies, Scrum, and Kanban	Apply	10	To assess students' ability to apply knowledge of agile methodologies in software development.
(FA)-3	Report	Create a project proposal document for a hypothetical software application	Apply	10	Simulates real-world tasks of planning and proposing software projects.

CIA - II

(FA)-4	Group Discussion	Participate in a discussion on the benefits and challenges of Extreme Programming (XP)	Analyze	10	Enhances critical thinking and the ability to evaluate and discuss software development methodologies.
(FA)-5	GitHub Repository Analysis	Analyze a GitHub repository for a popular open-source project. Submit a report detailing the structure, commits, and contributions.	Analyze	10	Sharpens analytical skills and understanding of collaborative software development.
(FA)-6	Presentation	Create and present a simple static web page using HTML and CSS	Apply	5	To assess students' ability to design and implement web pages using HTML and CSS.

Continuous Practical Assessment

Component	Capstone	Details of FA	Bloom's Level	Marks	Rationale
(CPA)-1	Lab Exercises	Complete weekly lab exercises on HTML, CSS, JavaScript, and Bootstrap	Apply	10	Ensures continuous hands-on practice and application of course concepts.
(CPA)-2	Attendance and Class Participation	Active participation in lectures and labs	Apply	10	Encourages regular attendance and engagement in class activities.

End Semester Practical Examination

Component	Capstone	Details of FA	Bloom's Level	Marks	Rationale
(SA)-1	Case Study	Develop a mini-project implementing a dynamic web page with interactive elements using HTML, CSS, and JavaScript	Analyze	10	Evaluates students' ability to integrate and apply their knowledge to solve a practical problem.
(SA)-2	Technical Skills	Assessment of skills in HTML, CSS, JavaScript, and GitHub through a practical test	Remember and Understand	20	Assesses the students' technical proficiency and understanding of key concepts.

U N I T - 1

To Learn Various Software Development Models.

Software Development Models

- In today's fast-paced digital landscape, the ability to develop high-quality software efficiently and effectively is crucial. This presentation will provide an in-depth exploration of the various software development models, from traditional approaches to modern agile methodologies and DevOps practices. By understanding the nuances of each model, you will gain the knowledge to choose the right approach for your software projects and deliver exceptional results.

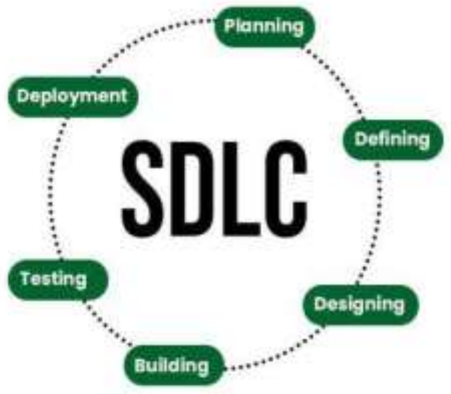
LEARNING OBJECTIVES

Introduction to software development process, Software engineering principles, requirements gathering and analysis, Overview of programming languages. Agile Software Development: Traditional Software Development Models, Agile methodology (Scrum, Kanban), DevOps practices, Comparison of methodologies.

Introduction to Software Development Process

Key Phases:

- Requirement Analysis: Identify and document software needs.
- Design: Plan the system architecture and components.
- Implementation: Write and compile the source code.
- Testing: Verify the software functions correctly.
- Deployment: Release the software to users.
- Maintenance: Update and fix software post-deployment.



SDLC Importance:

- Ensures Quality: Maintains high standards through structured phases.
- Provides Structure: Offers a clear framework for development.

Software Engineering Principles

Fundamental Principles:

- Modularity: Divide software into manageable parts.
- Abstraction: Simplify complex systems by focusing on essential details.
- Encapsulation: Bundle data and methods that operate on the data.
- Separation of Concerns: Divide a program into distinct sections.
- Cohesion: Ensure components are well-defined and related.
- Coupling: Minimize dependencies between components.

Impact on Software Development:

- Reliability: Produces dependable and predictable software.
- Maintainability: Facilitates easy updates and fixes.

Requirements Gathering and Analysis

The requirements gathering process for new features

3

Identify and discuss any corner cases.
"10% of users may meet _____ condition, and be unable to use the feature successfully. Correcting this will increase scope by 50% - how important is it?"

Write user stories for all required corner cases

Main Techniques:

- **Interviews:** Direct discussions to gather information.
- **Surveys:** Collect data from a large audience.
- **Document Analysis:** Review existing documentation.
- **Observation:** Watch users interact with the system.
- **Workshops:** Collaborative sessions to define requirements.

Importance:

- **Foundation for Development:** Establishes clear objectives.
- **Avoids Misunderstandings:** Ensures all stakeholders are aligned.

Overview of Programming Languages

Programming Example:

- **Procedural:** Step-by-step instructions (e.g., C).
- **Object-Oriented:** Objects and classes (e.g., Java).
- **Functional:** Functions and immutability (e.g., Haskell).
- **Scripting:** Automation and quick prototyping (e.g., Python).

Choosing a Language:

- **Project Requirements:** Match language capabilities to needs.
- **Performance Needs:** Ensure the language can handle the load.
- **Developer Expertise:** Utilize team strengths.

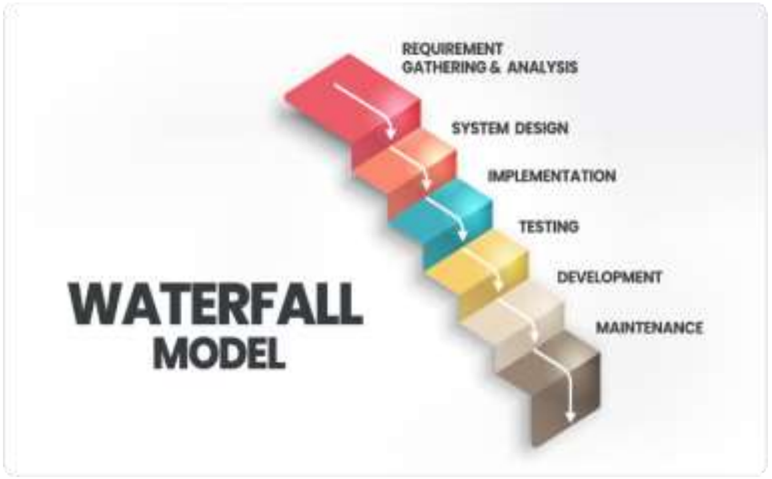
High-Level Languages: Easier to read and write, more abstract from hardware (e.g., Python, Java).

Low-Level Languages: Closer to hardware, more control over system resources (e.g., Assembly, Machine code).

Traditional Software Development Models

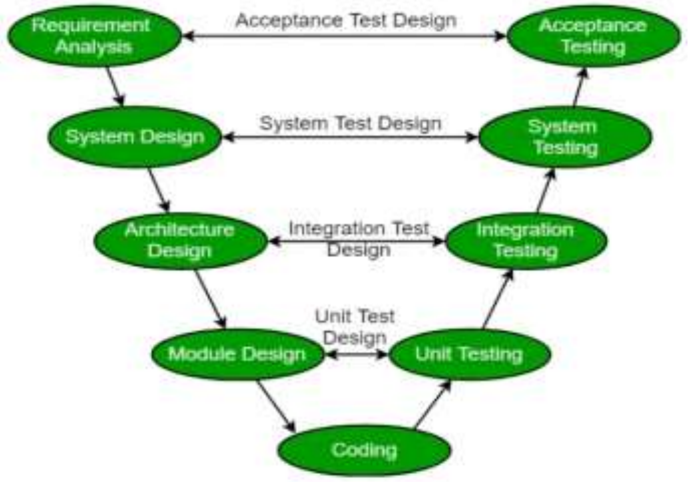
Waterfall Model

- The Waterfall model is a traditional, linear approach to software development, where each phase (requirements, design, development, testing, and deployment) is completed before the next phase begins. This model is well-suited for projects with well-defined and stable requirements, but it lacks the flexibility to adapt to changing needs.



V-Shaped Model

- The V-Shaped model is a variation of the Waterfall model, where testing activities are integrated throughout the development process. This approach helps to identify and address issues earlier, improving the overall quality of the software. However, it still maintains the linear, sequential nature of the Waterfall model.



Agile Software Development

Agile Principles

- Agile software development is an iterative and incremental approach that emphasizes collaboration, flexibility, and rapid delivery of working software. The core principles of Agile include responding to change, delivering working software frequently, and involving the customer throughout the development process.

Scrum Methodology

- Scrum is a popular Agile framework that uses sprints, cross-functional teams, and constant feedback to deliver working software in short iterations. Key roles in Scrum include the Product Owner, Scrum Master, and Development Team, while artifacts like the Product Backlog and Sprint Backlog help to manage the project.

Kanban Methodology

- Kanban is another Agile approach that focuses on continuous delivery and visualizing the workflow. It uses a Kanban board to track the progress of work items, emphasizing the "pull" system to limit work-in-progress and enhance efficiency.

DevOps Practices

Collaboration:

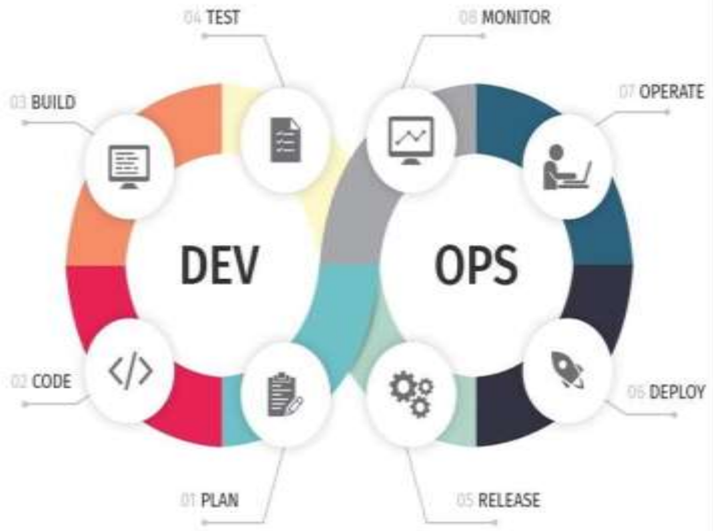
- DevOps emphasizes the collaboration between development and operations teams, breaking down the traditional silos and fostering a culture of shared responsibility and continuous improvement.

Automation

- DevOps practices rely heavily on automation to streamline the software delivery process, including tasks like building, testing, and deploying applications.

Continuous Integration and Delivery

- DevOps promotes the practice of Continuous Integration (CI) and Continuous Deployment (CD), where code changes are automatically built, tested, and deployed to production, ensuring a faster and more reliable release cycle.



Comparison of Methodologies

Agile vs. Waterfall vs. DevOps:

- Flexibility: Agile and DevOps are adaptable; Waterfall is static.
- Speed: Agile and DevOps promote rapid iterations; Waterfall is slower.
- Collaboration: Agile and DevOps emphasize teamwork; Waterfall is more isolated.

Best Use Cases:

- Agile: Suitable for projects with evolving requirements.
- Waterfall: Ideal for well-defined, unchanging projects.

Important Questions

- *What are the key phases in the software development process?*
- *How does the software development life cycle (SDLC) ensure the quality of a software product?*
- *What are the fundamental principles of software engineering?*
- *Why is requirements analysis crucial in the software development process?*
- *How do you choose the appropriate programming language for a specific project?*
- *How does Agile development differ from traditional software development models?*
- *What is DevOps, and how does it bridge the gap between development and operations?*

Case Study

- Simple application idea (e.g., calculator, to-do list)
- Technology stack chosen (programming language, frameworks)
- Basic functionalities to be implemented
- Demo of the developed application (optional)

References

Textbooks and Resources:

- RAJIB MALL - Fundamentals of software engineering
- Pressman, R. S. Software Engineering: A Practitioner's Approach
- Sommerville, I. Software Engineering
- Beck, K. Extreme Programming Explained: Embrace Change
- Online Articles and Guides