Sanjivani University
School of Engineering and Technology
Department of Computer Science and Engineering (AIML)

Application Development Practices

(24UMLES101)

Unit -1

Software Development Model

By,
Mr. Yogesh Sopan Modhe

Contents

- Introduction to software development process
- Software Engineering Principles
- Requirement Gathering and Analysis
- Traditional Software Development Models
- Overview of Programming Languages
- Agile Software Development
- Agile Methodology (Scrum, Kanban)
- DevOps practices,
- Comparison of Methodologies
- Case study

Software ?

- What is Software ? - Software is a set of instructions, data or programs used to operate computers and execute specific tasks. It is the opposite of hardware, which describes the physical aspects of a computer. Software is a generic term used to refer to applications, scripts and programs that run on a device.
- Software is defined as a collection of programs, documentation and operating procedures. **The Institute of Electrical and Electronic Engineers (IEEE)** defines software as a 'collection of computer programs, procedures, rules and associated documentation and data.
- Why it is essential ? –
    - For Automation
    - Data Management and Analysis
    - Improve User Experience
    - Innovation and Problem Solving
    - Global Connectivity
    - Enhanced Communication etc.

Software Development Process

- Process – It's essential as it's productive, time-saving, and helps us in fast delivery and development with more accuracy.
- A generic process framework for software engineering encompasses five activities:
  - ✓ Communication
  - ✓ Planning
  - ✓ Modeling
  - ✓ Construction
  - ✓ Deployment
- The essence of practice –
  - ✓ Understand the problem (communication and analysis)
  - ✓ Plan a solution (modeling and software design)
  - ✓ Carry out the plan (code generation)
  - ✓ Examine the result for accuracy (testing and quality assurance).

Software Development Process

Fig 1. Software Development Process

Software Engineering Principles

- The Reason it all exists
- Keep It Simple
- Maintain the Vision
- What You Produce, Others Will Consume
- Be Open to the Future
- Plan Ahead for Reuse
- Think

Requirement Gathering and Analysis

- In software development, the requirements gathering and analysis process is typically led by a dedicated team, usually consisting of business analysts, project managers, and occasionally lead developers. Their role is to systematically uncover and define the needs and conditions for the software project, ensuring alignment with business goals and user requirements.
- This development team engages with various stakeholders, which may include customers, end users, and internal team members, to gain comprehensive insight into what the software must do. They use methods such as interviews, surveys, workshops, and document analysis to gather detailed information about user needs, business processes, business process documentation, and system requirements.
- At this stage, they may decide to develop a proof of concept project to test the feasibility of the initial assumptions – a key step in developing non-standard software solutions.
- This team plays a critical role in bridging the gap between the technical and business aspects of software development.

Software Development Process Models

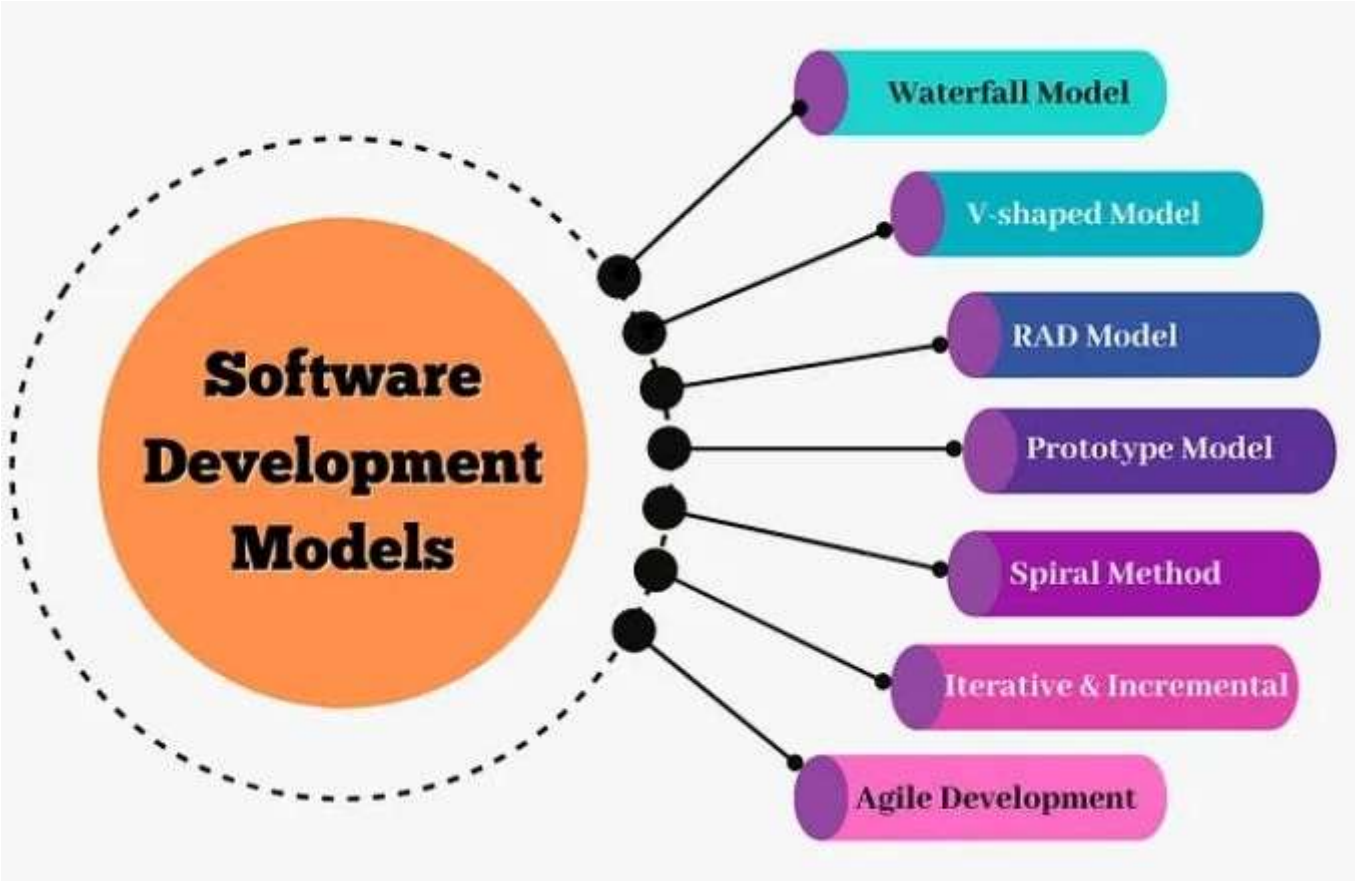


Fig. 2 Various software development models

Waterfall Model

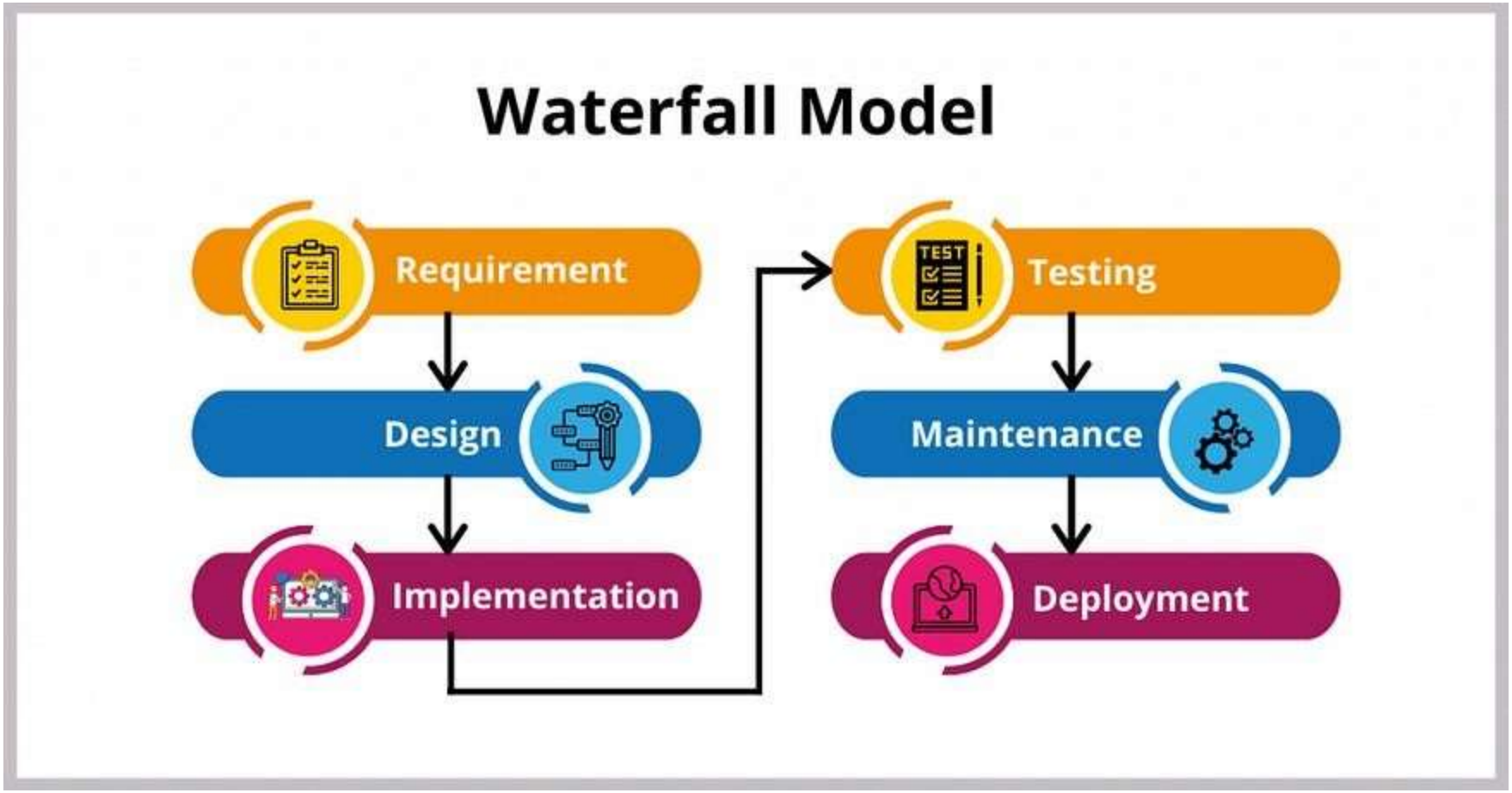


Fig. 3 Waterfall Model

Waterfall Model

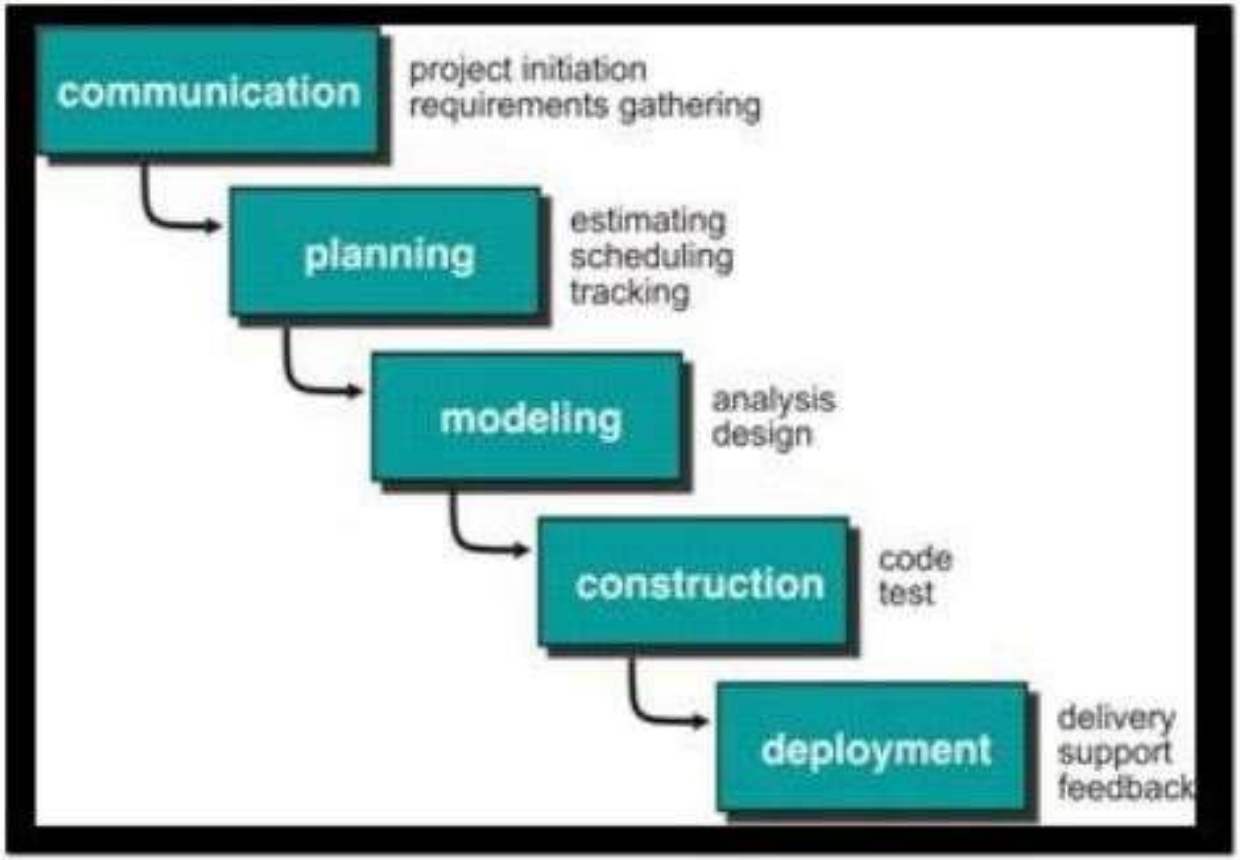


Fig. 3.1 Waterfall Model

V-Shaped Model

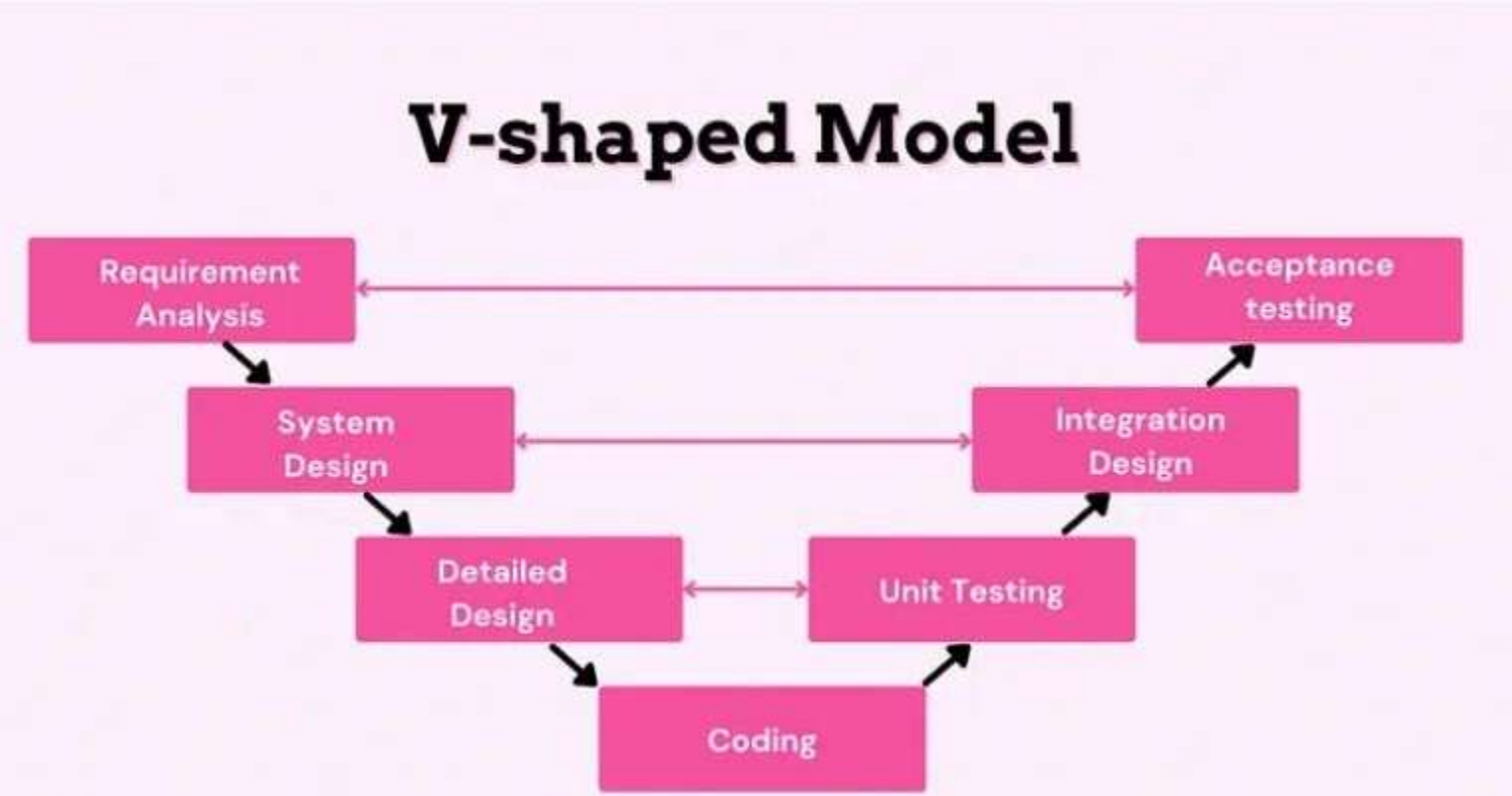


Fig. 4 V-Shaped Model

Rapid Application Development (RAD) Model

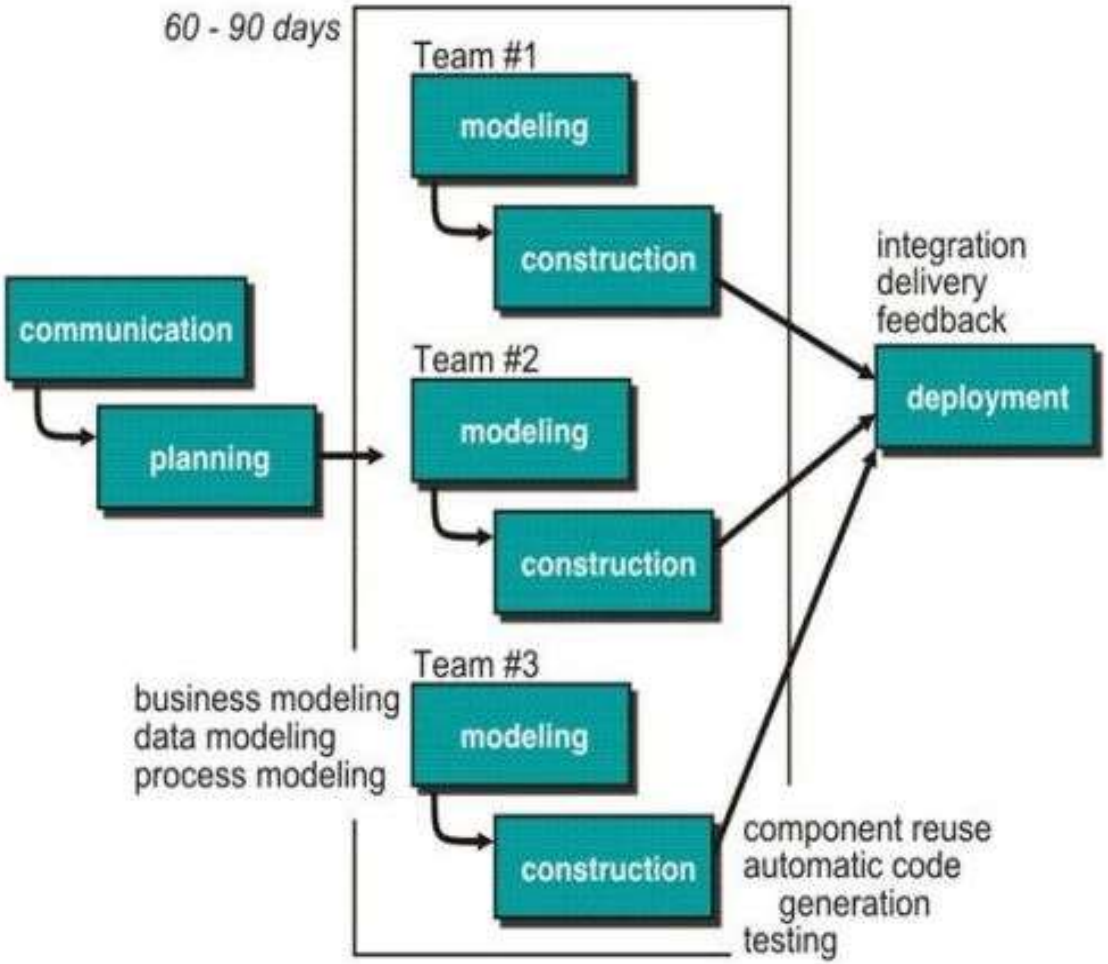
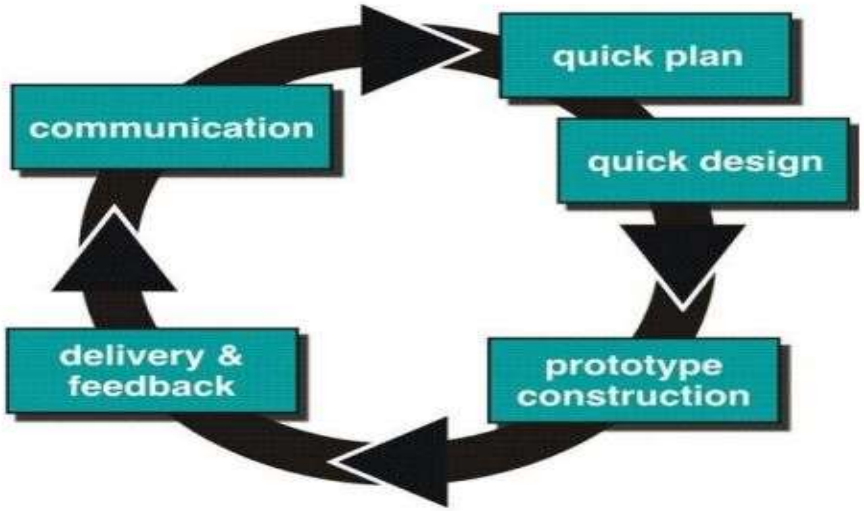


Fig. 5 RAD Model

Prototype Model

- **Situation in which prototyping model is applicable**: Often, a customer defines a set of general objectives for software, but does not identify detailed requirements for functions and features. In other cases, the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human-machine interaction should take. In these, and many other situations, a prototyping paradigm may offer the best approach.



Spiral Model

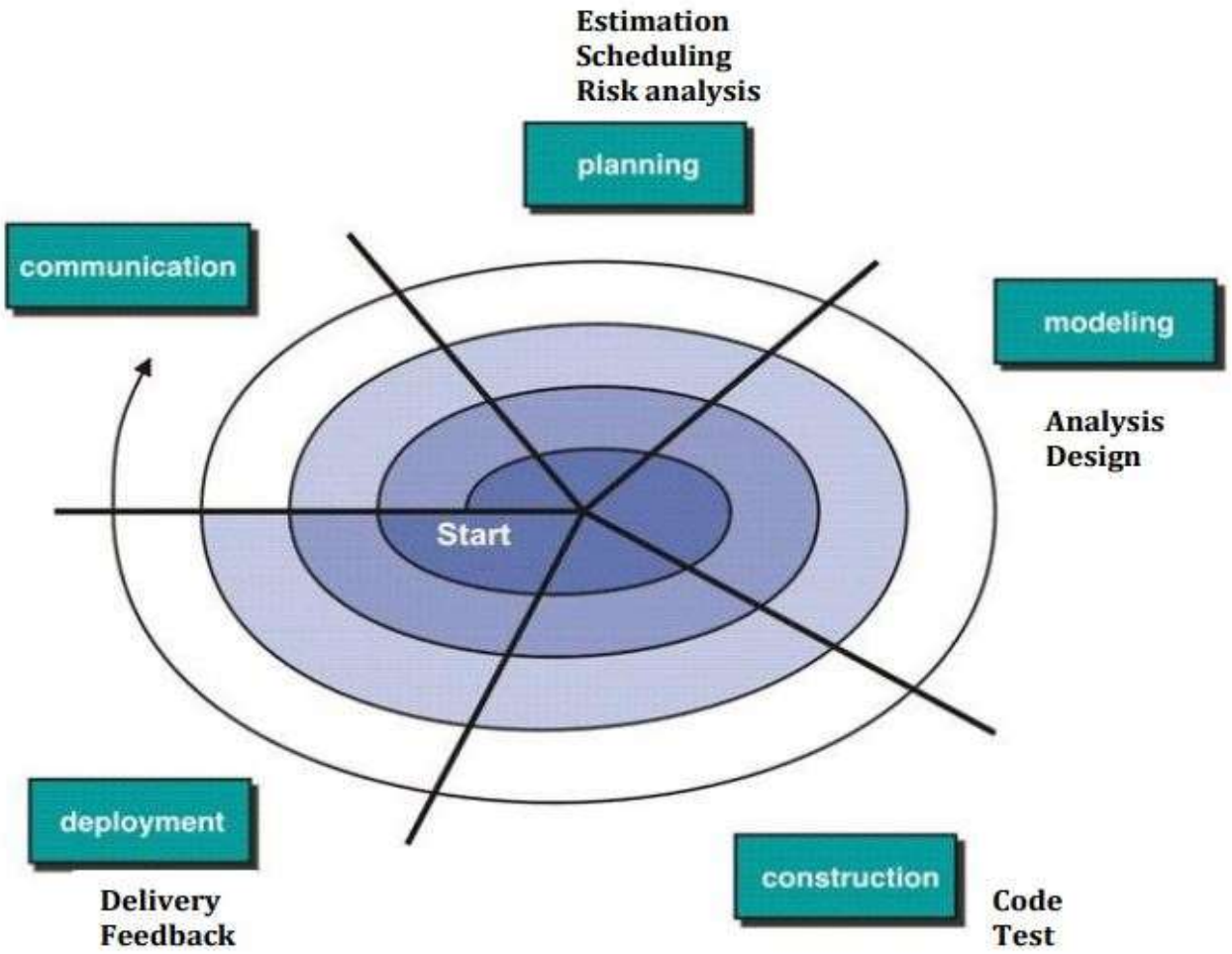
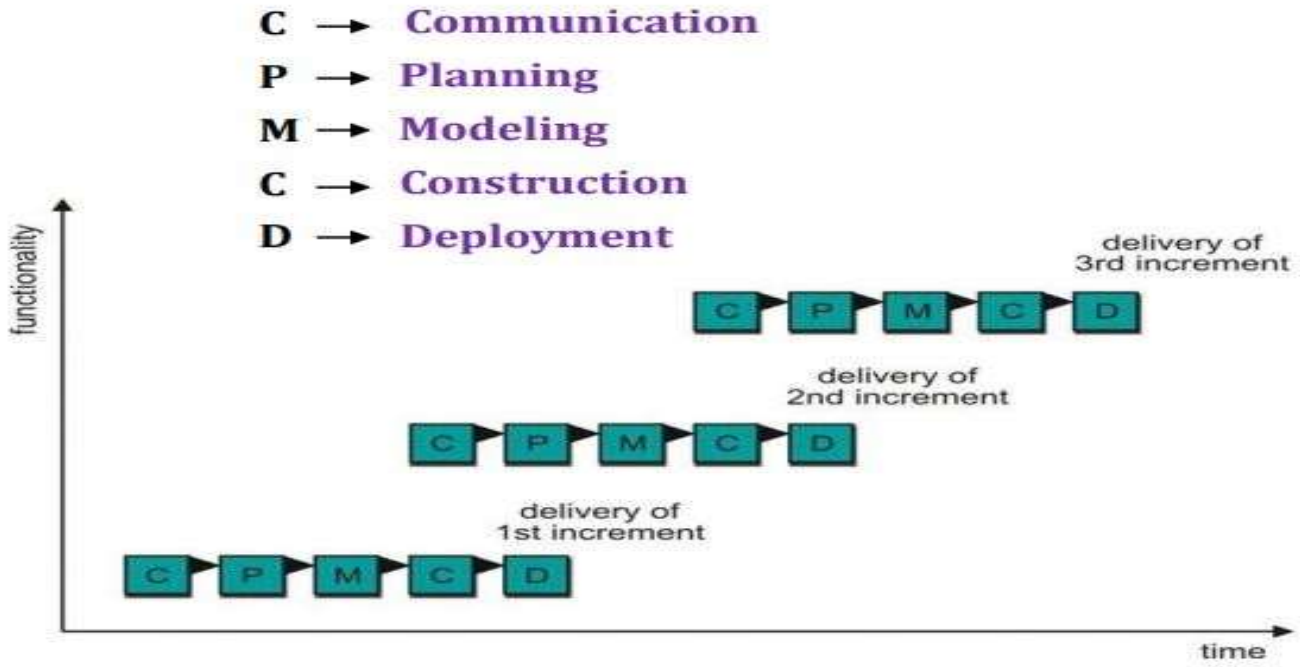


Fig. 6 Spiral Model

Iterative and Incremental Model

- The incremental model allows stakeholders and developers to check results using the first increment. The developer only focuses on a clear and complete definition of the whole system before you start. The Model is great for a project with loosely coupled parts and projects with exact, precise needs.



Overview of Programming Languages

- A programming language is a set of instructions and syntax used to create software programs. Some of the key features of programming languages include:
  - Syntax: The specific rules and structure used to write code in a programming language.
  - Data Types: The type of values that can be stored in a program, such as numbers, strings, and booleans.
  - Variables: Named memory locations that can store values.
  - Operators: Symbols used to perform operations on values, such as addition, subtraction, and comparison.
  - Control Structures: Statements used to control the flow of a program, such as if-else statements, loops, and function calls.
  - Libraries and Frameworks: Collections of pre-written code that can be used to perform common tasks and speed up development.
  - Paradigms: The programming style or philosophy used in the language, such as procedural, object-oriented, or functional.

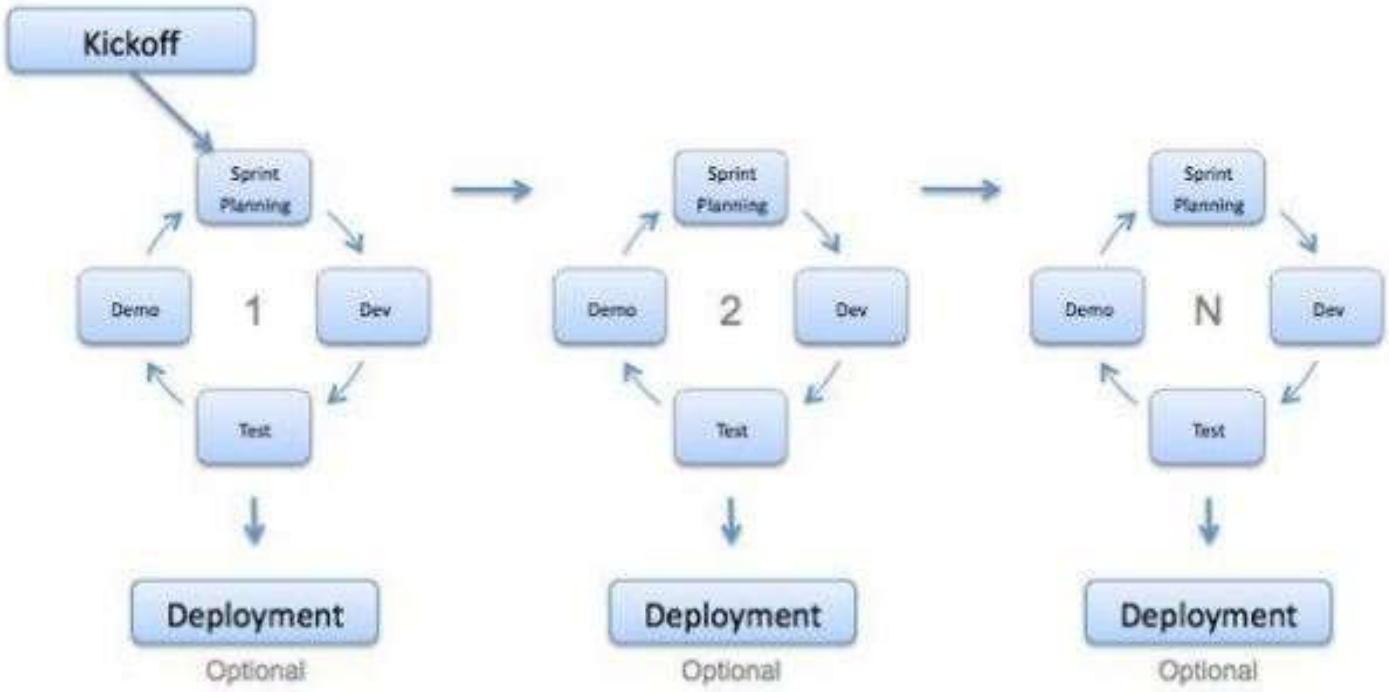
Agile Software Development

- **What is Agile ?-** The Agile methodology is a project management approach that involves breaking the project into phases and emphasizes continuous collaboration and improvement. Teams follow a cycle of planning, executing, and evaluating.
- https://www.youtube.com/watch?v=8eVXTyIZ1Hs
- Agility Principles

Priority is to satisfy the customer through early and continuous delivery of valuable software	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	Business people and developers must work together daily throughout the project.
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
Working software is the primary measure of progress.	Agile processes Promote sustainable development
Continuous attention to technical excellence and good design enhances agility.	Simplicity—the art of maximizing the amount of work not done—is essential.
The best architectures, requirements, and designs emerge from self– organizing teams.	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Software Development

- Agile Software Development sample case study with - https://asana.com/templates/agile-project-plan



Agile Software Development

- **Advantages of Agile model:**

  - Customer satisfaction by rapid, continuous delivery of useful software.
  - People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
  - Working software is delivered frequently (weeks rather than months).
  - Face-to-face conversation is the best form of communication.
  - Close, daily cooperation between business people and developers.
  - Continuous attention to technical excellence and good design.
  - Regular adaptation to changing circumstances.
  - Even late changes in requirements are welcomed

Agile Software Development

- **Disadvantages of Agile model:**
  - In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
  - There is lack of emphasis on necessary designing and documentation.
  - The project can easily get taken off track if the customer representative is not clear what final outcome that they want

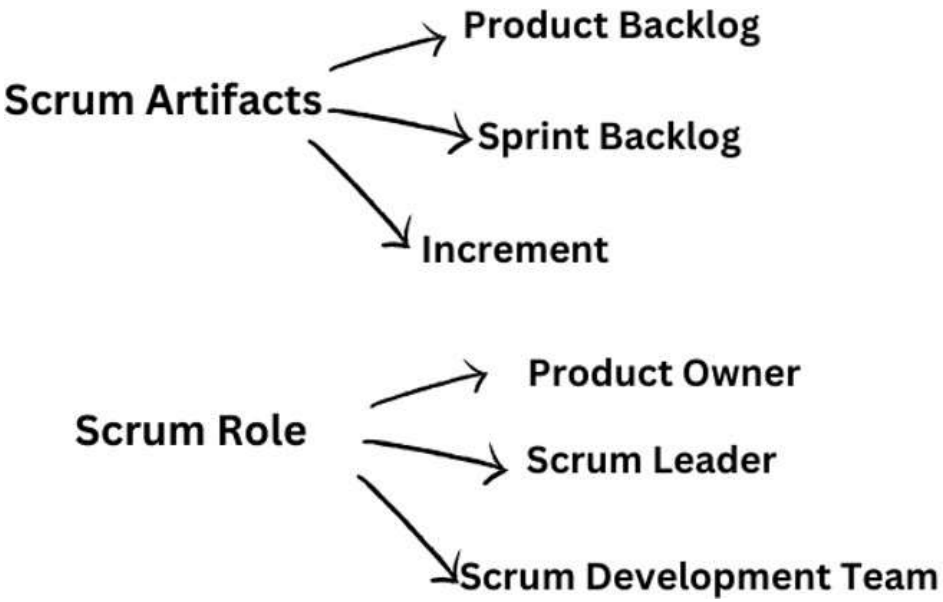
Scrum

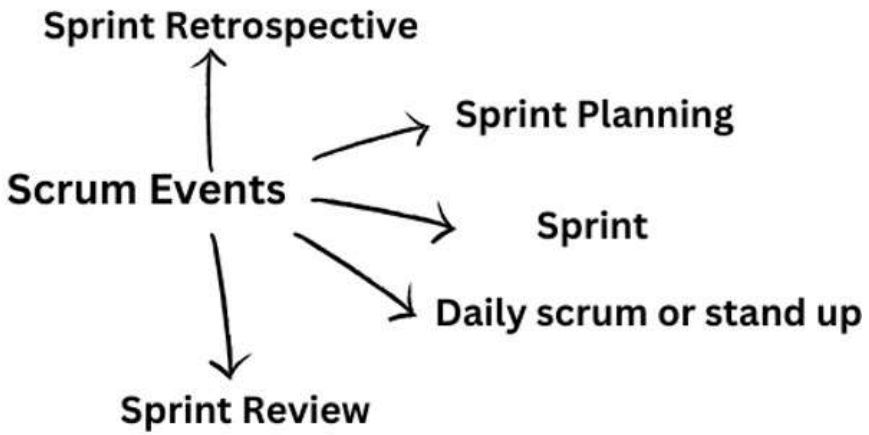
- Scrum is a management framework that teams use to self-organize and work towards a common goal. It describes a set of meetings, tools, and roles for efficient project delivery.
- Scrum Principles
  - Transparency
  - Reflection
  - Adaption
- Scrum values for Project team-
  - Commitment
  - Courage
  - Focus
  - Openness
  - Respect

Source- https://aws.amazon.com/what-is/scrum/

Working of Scrum

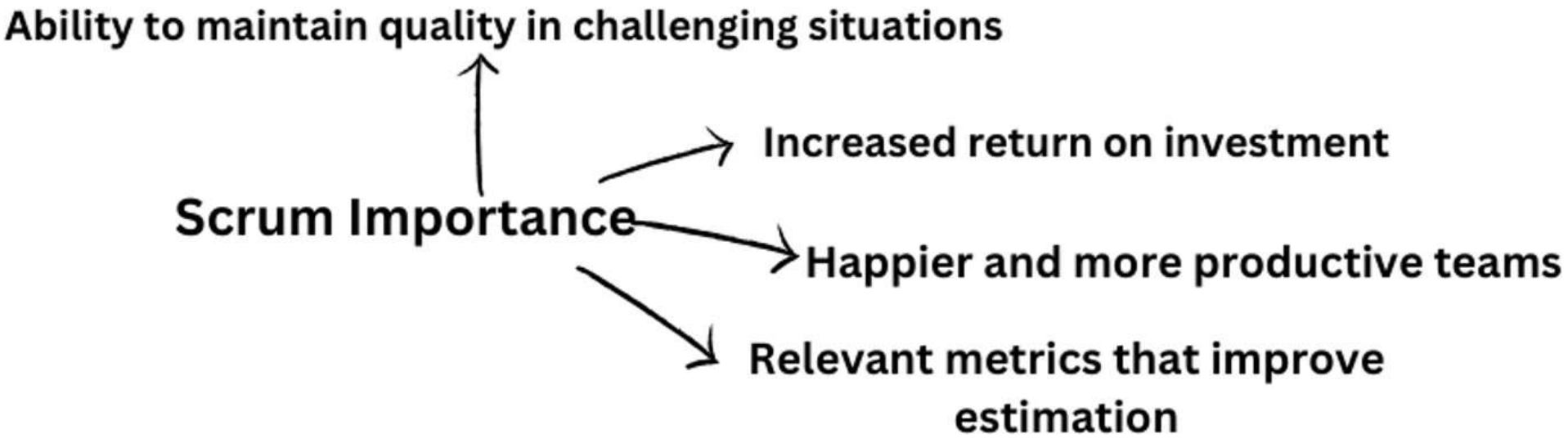
- How does it work ? - Scrum is a framework that is easy to learn but difficult to become an expert in. The essence of Scrum is a self-organizing team delivering customer value in a time-boxed period called a Sprint. Scrum defines artifacts, roles, and events associated with each Sprint.





Source- https://aws.amazon.com/what-is/scrum/

Importance of Scrum in S/W Development



Source- https://aws.amazon.com/what-is/scrum/

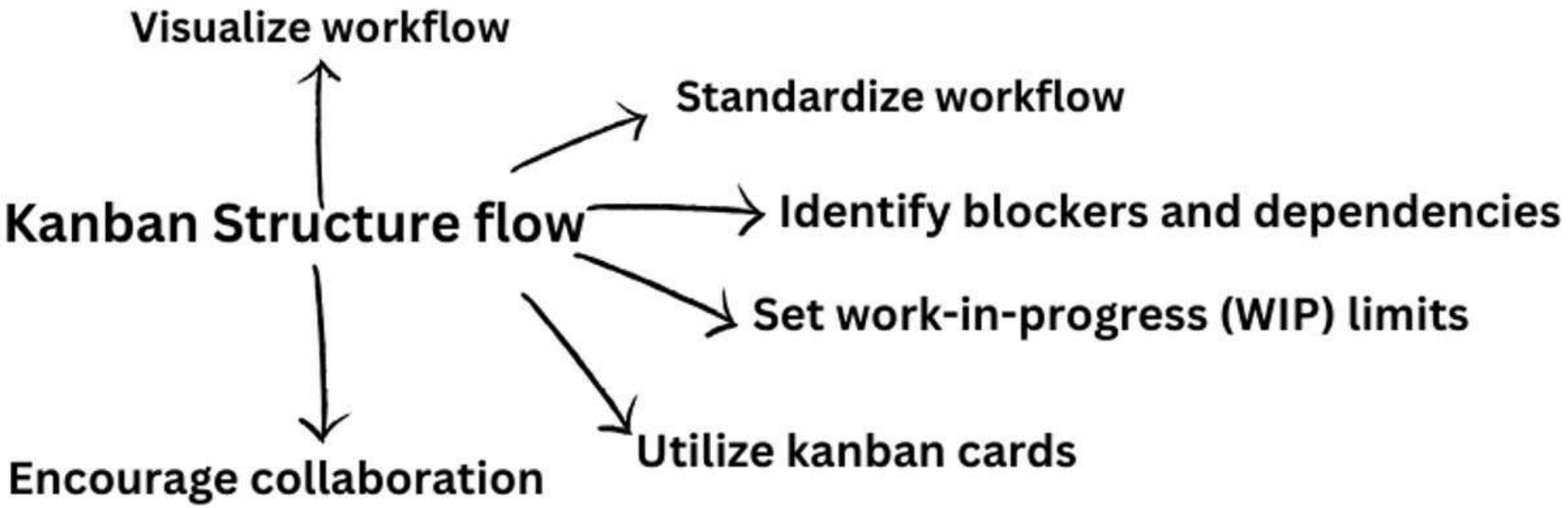
Scrum vs. Agile

- Agile refers to a mindset or way of thinking in software development.
- It is a philosophy adopted at an organizational level to get every team member to focus on continuous improvement and value delivery to customers.
- Scrum is a framework for getting work done within agile.
- Scrum uses all the core principles of agile to define methods to facilitate a project.
- However, it is important to note that agile does not always mean Scrum.

Source- https://aws.amazon.com/what-is/scrum/

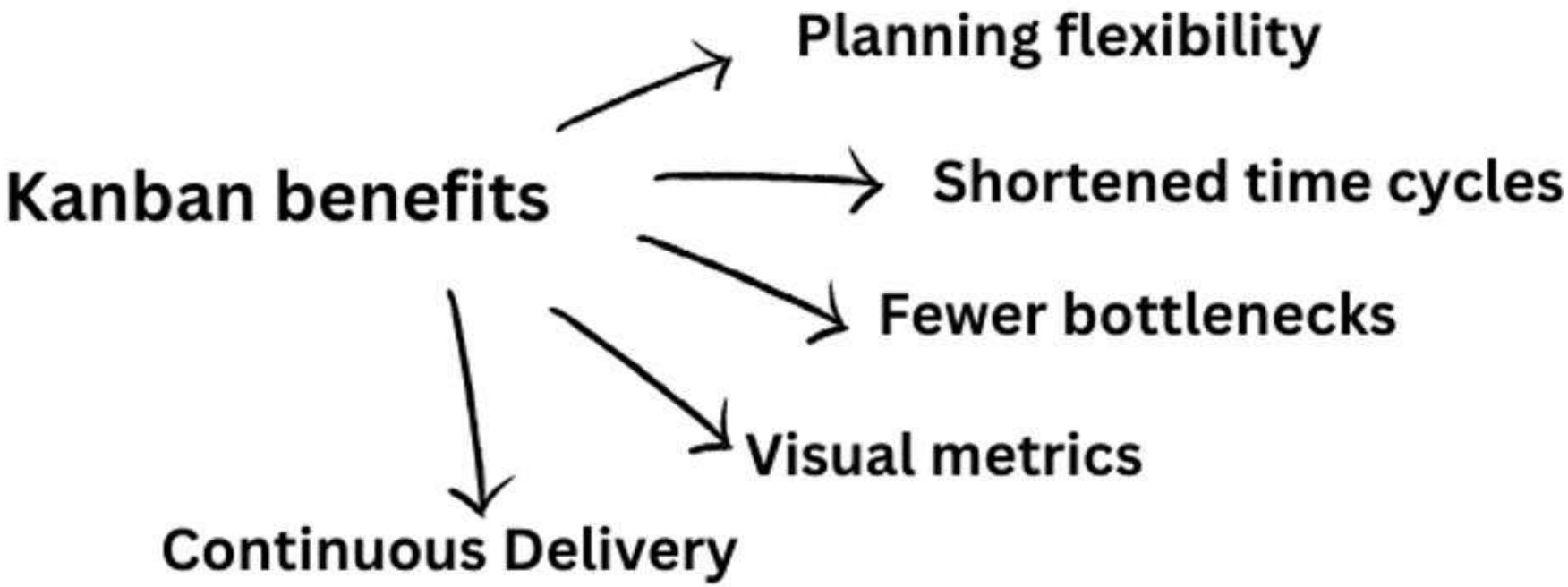
Kanban

- Kanban is a popular framework used to implement Agile and DevOps software development. It requires real-time communication of capacity and full transparency of work. Work items are represented visually on a kanban board, allowing team members to see the state of every piece of work at any time.



Source- https://www.atlassian.com/agile/kanban

Benefits of Kanban



Source- https://www.atlassian.com/agile/kanban

Scrum vs. Kanban

	SCRUM	KANBAN
Release methodology	Regular fixed-length sprints (i.e., two weeks)	Continuous flow
Roles	Product owner, scrum master, development team	Continuous delivery or at the team's discretion
Key metrics	Velocity	Cycle time
Change philosophy	Teams should strive not to change the sprint forecast during the sprint. Doing so compromises learning around estimation.	Change can happen at any time

DevOps

- DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.

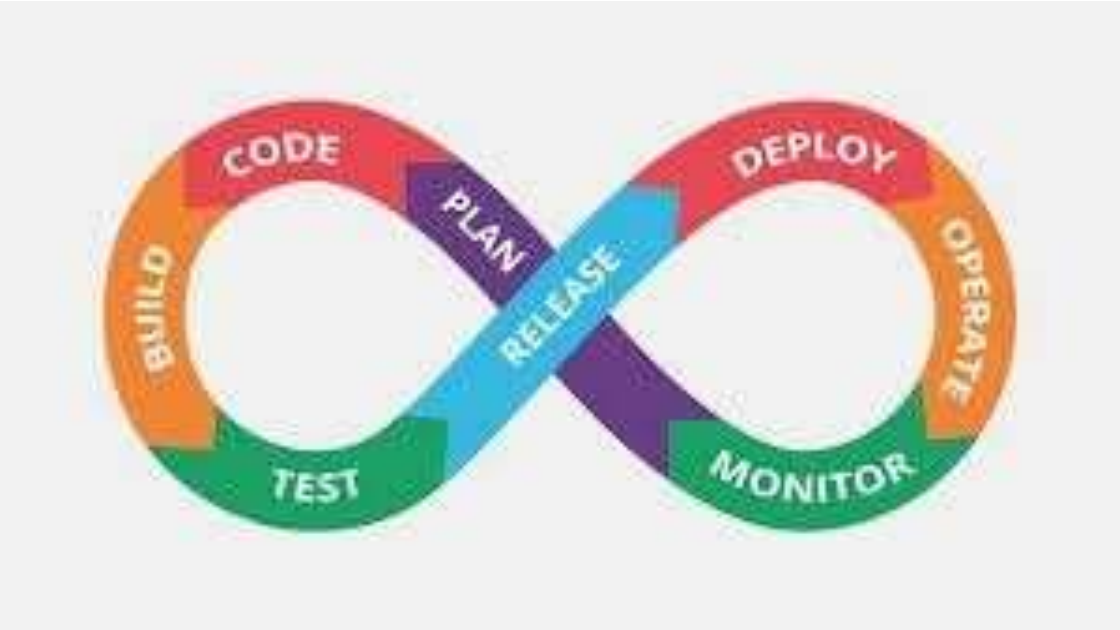
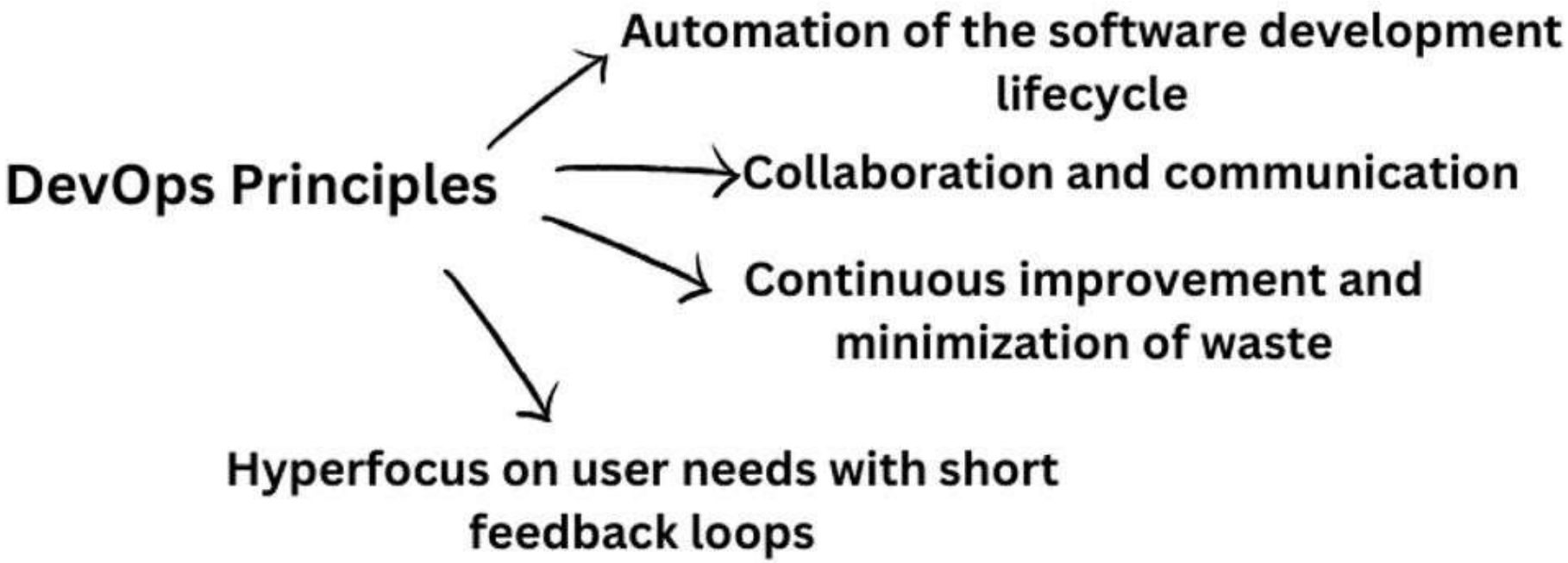


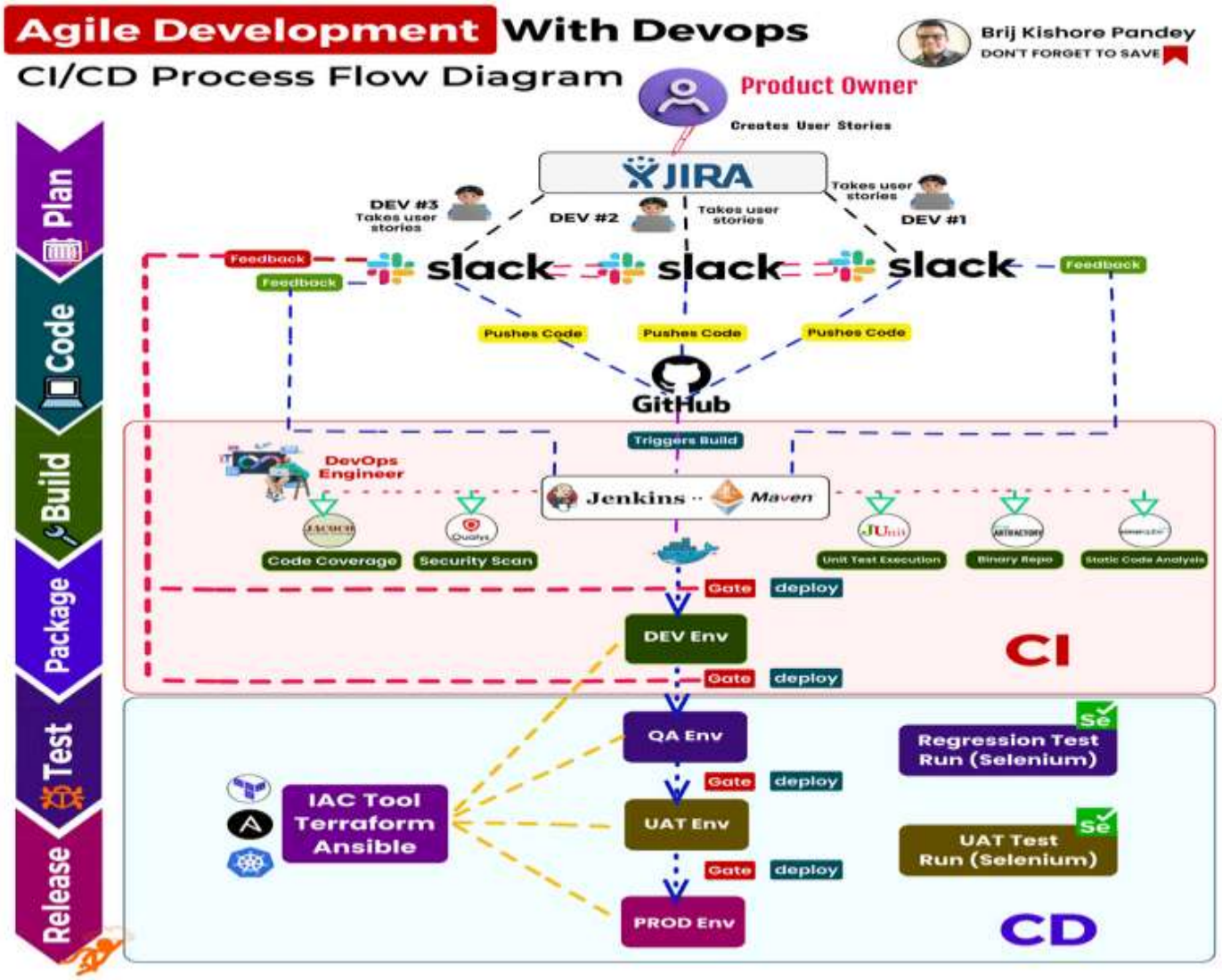
Fig. 7 DevOps Process

Core DevOps Principles



Source- https://about.gitlab.com/topics/devops/

Core DevOps Principles



DevOps Phases

- Phase 1: Bring Your Own DevOps (BYOD)
- Phase 2: Best-in-class DevOps
- Phase 3: Do-it-yourself (DIY) DevOps
- Phase 4: DevOps Platform

DevOps tools, concepts and fundamentals

Topic	Description
Version control	The fundamental practice of tracking and managing every change made to source code and other files. Version control is closely related to source code management.
Agile	Agile development means taking iterative, incremental, and lean approaches to streamline and accelerate the delivery of projects.
Continuous Integration (CI)	The practice of regularly integrating all code changes into the main branch, automatically testing each change, and automatically kicking off a build.
Continuous Delivery (CD)	Continuous delivery works in conjunction with continuous integration to automate the infrastructure provisioning and application release process. They are commonly referred to together as CI/CD.
Shift left	A term for shifting security and testing much earlier in the development process. Doing this can help speed up development while simultaneously improving code quality.

Benefits of DevOps

- Collaboration - Adopting a DevOps model creates alignment between development and operations teams
- Fluid responsiveness - More collaboration leads to real-time feedback and greater efficiency; changes and improvements can be implemented quicker and guesswork is removed
- Shorter cycle time - Improved efficiency and frequent communication between teams shortens cycle time; new code can be released more rapidly while maintaining quality and security.

Comparisons

- Case study to find the comparison between Agile, Scrum, Kanban and DevOps.

References

1Rogers Pressman, “Software Engineering- A Practitioner’s Approach”, 7th Edition, McGraw Hill Edition. Chapter 1-3 from page number 1-65.

2 Scrum - https://aws.amazon.com/what-is/scrum/

3 Kanban - https://www.atlassian.com/agile/kanban

4 DevOps- https://about.gitlab.com/topics/devops/