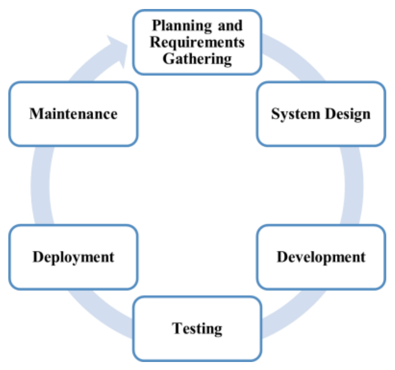
BUSINESS ANALYST

A **business analyst** is a professional who works with organizations to identify business needs, problems, and opportunities for improvement. They gather, analyze, and document business requirements, processes, and systems to help create solutions that improve efficiency, productivity, and overall business performance. They often act as a bridge between stakeholders (such as management, users, and IT teams) to ensure that business goals are aligned with technical capabilities.

SDLC (Software Development Life Cycle)

SDLC (Software Development Life Cycle) is a structured approach to software development that outlines the process for planning, designing, developing, testing, deploying, and maintaining software systems. It helps in ensuring the development of high-quality software that meets user requirements and is delivered on time and within budget.

The typical stages of SDLC include:



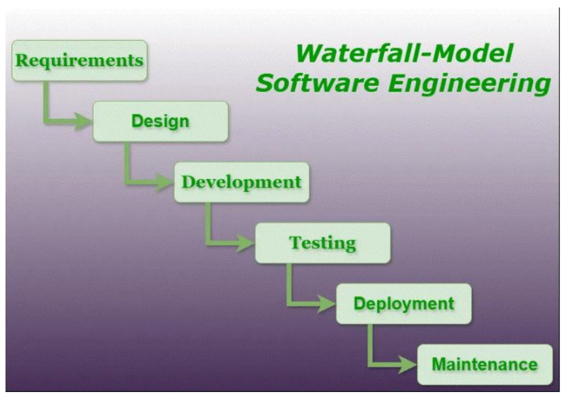
1. **Planning and Requirements Gathering**: Understanding the business needs and defining the system's requirements.
2. **System Design**: Creating the architecture and design specifications for the system.
3. **Development**: Coding and building the system.
4. **Testing**: Identifying and fixing bugs and ensuring the system works as intended.
5. **Deployment**: Releasing the system to the users.
6. **Maintenance**: Updating the system and fixing any future issues.

Why it comes under Business Analysis: Business analysts play a key role in the SDLC, especially in the initial stages of planning and requirements gathering. They work with stakeholders to understand business needs and translate them into clear, actionable requirements that can guide the development process. By ensuring that the software aligns with business objectives, a business analyst helps ensure the successful implementation of the system.

There are several methodologies that come under the **Software Development Life Cycle (SDLC)**, each with its own approach to managing the development process. Here are the most common ones:

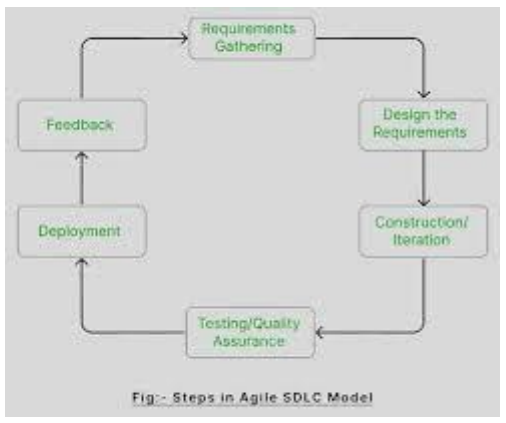
1. Waterfall Model

- **Approach**: Linear and sequential. Each phase must be completed before the next phase begins.
- **Phases**: Requirements gathering → Design → Development → Testing → Deployment → Maintenance.
- **Best for**: Projects with well-defined requirements that are unlikely to change.



2. Agile Model

- **Approach**: Iterative and incremental. Development is done in small, manageable sections called sprints (usually 2-4 weeks).
- **Phases**: Planning → Design → Development → Testing → Deployment → Feedback (repeated in cycles).
- **Best for**: Projects with evolving requirements and frequent changes.



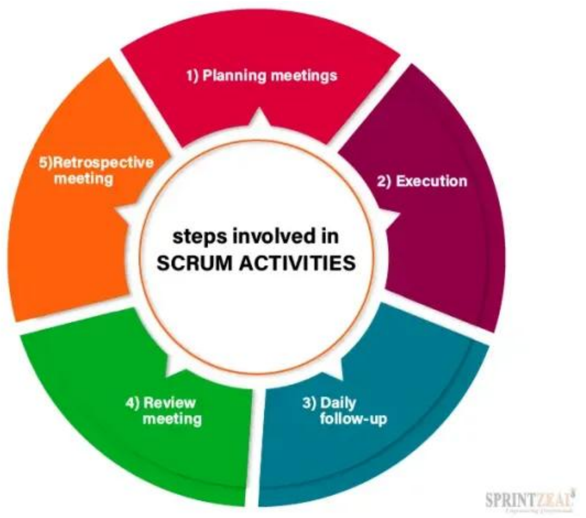
Key Concepts in Agile:

- **Product Backlog**: The product backlog is a prioritized list of features, enhancements, fixes, and technical tasks required for a product. It serves as the primary source of work items for the development team and is constantly refined and updated by the Product Owner. The backlog ensures that the most important tasks are addressed first to deliver value to users.
- **Sprint Planning**: Sprint planning is a meeting that occurs at the beginning of each sprint in Agile. The team selects a set of prioritized tasks from the product backlog to work on during the sprint. The goal is to define clear objectives and break down user stories into smaller, actionable tasks. Sprint planning helps ensure that the team is aligned on what needs to be delivered in the short cycle.
- **User Stories**: User stories are concise, user-centered descriptions of features or functionality in the product. Written from the perspective of the end user, they follow a format like: "As a [user], I want [goal] so that [benefit]." User stories are part of the product backlog and help ensure that development focuses on delivering value to users. They are broken into smaller tasks during sprint planning.
- **Jira Board**: A visual tool used in Agile projects to manage tasks and track progress. The board displays the status of tasks (user stories, bugs, or technical tasks) across various columns such as "To Do", "In Progress", and "Done." It provides

transparency and helps teams collaborate effectively by visualizing the workflow.
Note: While Jira is primarily designed for **Agile** methodologies (like **Scrum** and **Kanban**), it can also be adapted for other project management approaches such as **Waterfall** and **DevOps**. Teams can customize Jira workflows and use it to track tasks and progress in a sequential or continuous deployment environment.

3. Scrum (Agile Framework)

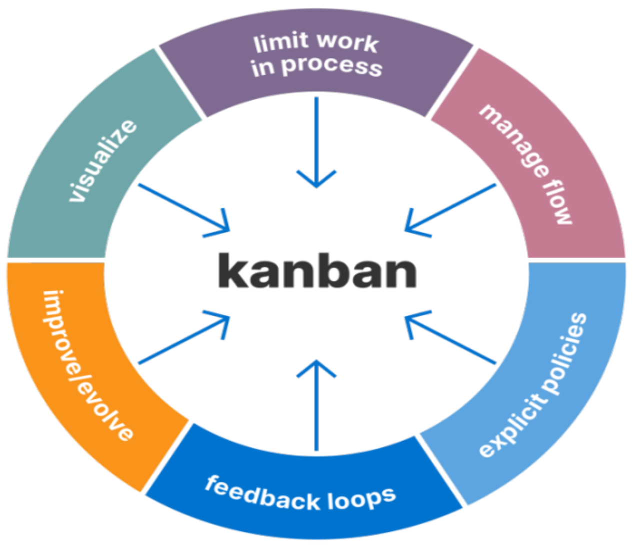
- **Approach**: A subset of Agile, Scrum focuses on time-boxed iterations (sprints) and regular updates.
- **Phases**: Product Backlog → Sprint Planning → Sprint → Daily Standup → Sprint Review → Sprint Retrospective.
- **Best for**: Teams that need structure in Agile development and frequent collaboration.

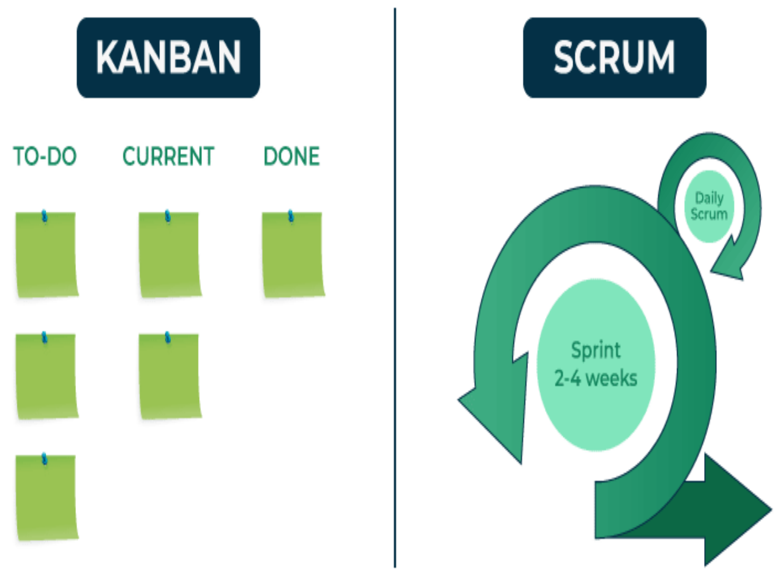


4. Kanban (Agile Framework)

- **Approach**: Kanban focuses on visualizing workflows, limiting work in progress (WIP), and improving efficiency. It uses a Kanban Board to track tasks as they move through stages such as "To Do," "In Progress," and "Done."
- **Phases**: Unlike Scrum, Kanban does not follow strict phases or sprints. Work items are continuously pulled from a backlog based on capacity.
- **Key Principles**:
    1. Visualize the workflow.
    2. Limit work in progress (WIP).
    3. Focus on continuous delivery.
    4. Improve collaboratively.
- **Best for**: Projects requiring flexibility, continuous delivery, and teams focusing on improving existing workflows rather than delivering in iterations.

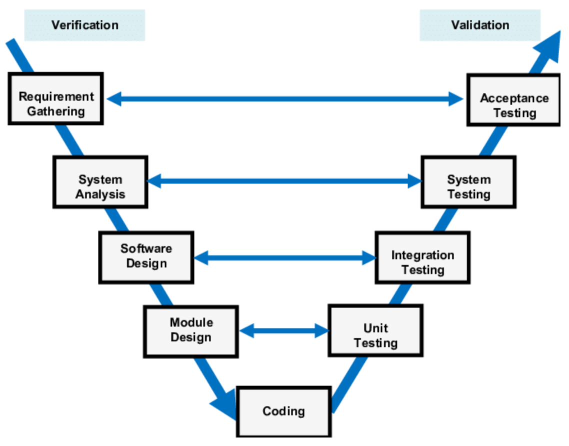
Example: A support team using Kanban can manage incoming tickets by visualizing their statuses (e.g., "New," "In Progress," "Resolved") and ensuring no team member is overwhelmed with too many tickets at once.





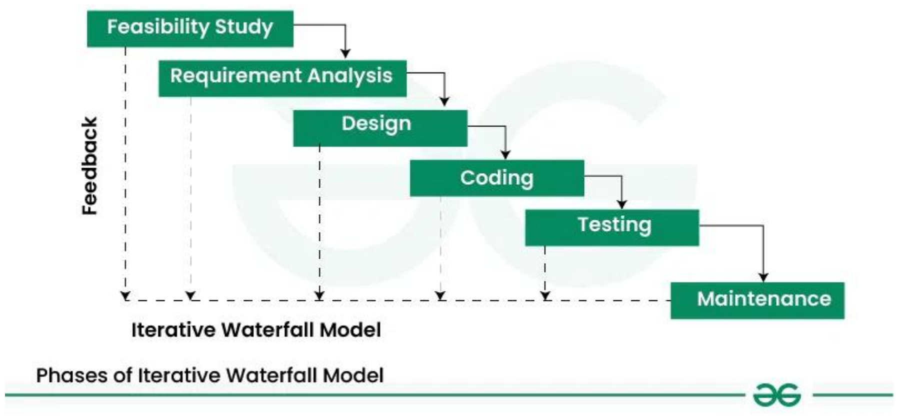
5. V-Model (Verification and Validation Model)

- **Approach**: A sequential model similar to Waterfall but with a focus on validation and verification during each phase.
- **Phases**: Requirements Analysis → System Design → Coding → Unit Testing → Integration Testing → System Testing → Deployment.
- **Best for**: Projects where testing and quality are critical.



6. Iterative Model

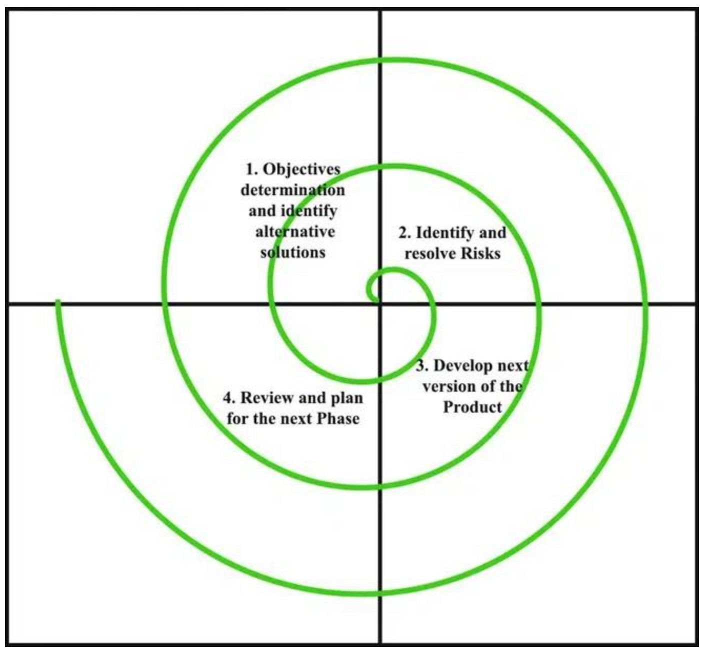
- **Approach**: Development is done in iterations, with each iteration delivering a portion of the product.
- **Phases**: Planning → Design → Development → Testing → Evaluation → Repeat.
- **Best for**: Projects where the full requirements are unclear and evolve over time.



Phases of Iterative Waterfall Model

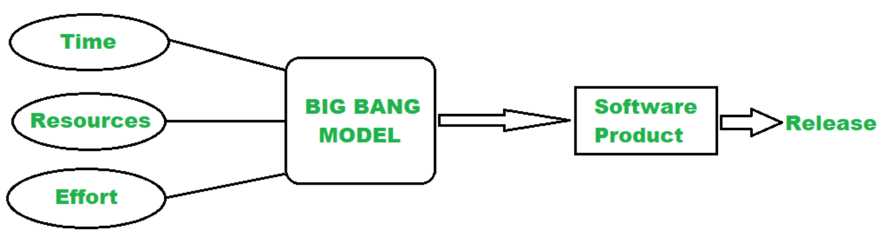
7. Spiral Model

- **Approach**: Combines elements of both design and prototyping with an emphasis on risk assessment. It involves continuous refinement through iterative cycles.
- **Phases**: Planning → Risk Analysis → Engineering → Testing → Evaluation → Repeat.
- **Best for**: Complex and large-scale projects with high risks.


1. Objectives determination and identify alternative solutions
2. Identify and resolve Risks
3. Develop next version of the Product
4. Review and plan for the next Phase

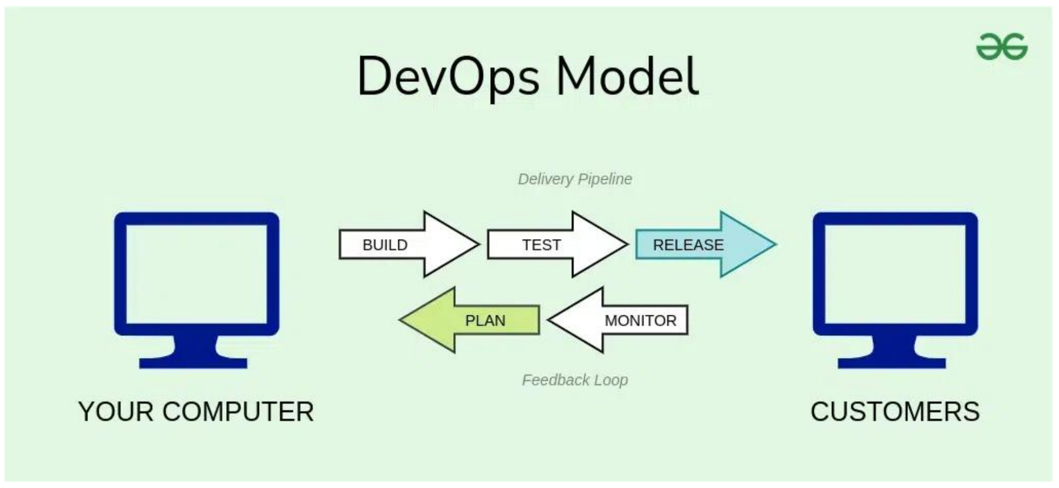
8. Big Bang Model

- **Approach**: The development process starts with no detailed planning, and all work is done simultaneously. The final product is built in one go.
- **Phases**: All stages are executed together, without formal phase division.
- **Best for**: Small projects or experimental projects with undefined requirements.



9. DevOps Model

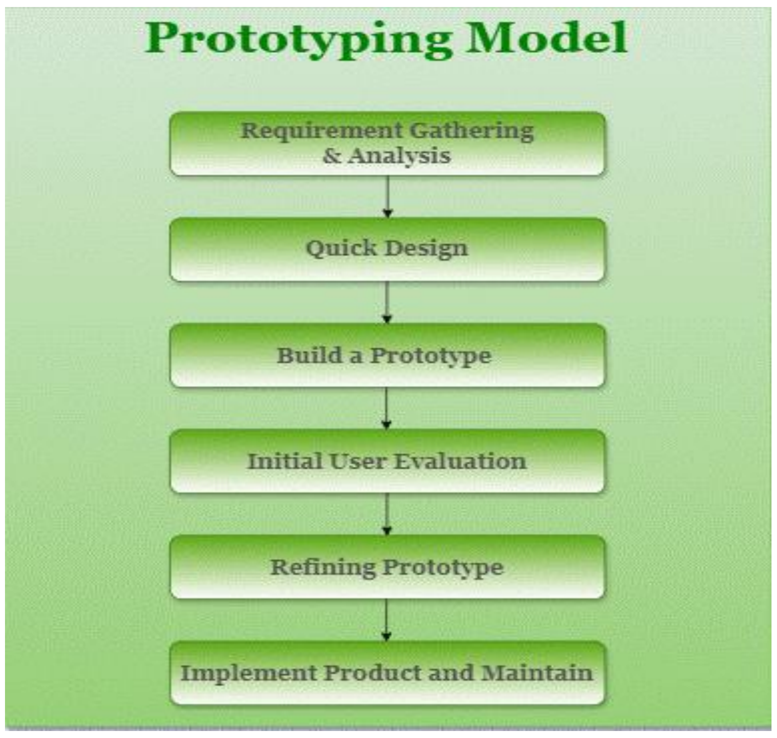
- **Approach**: Focuses on collaboration between development and operations teams to deliver software more quickly and reliably. It emphasizes automation, continuous integration, and continuous delivery (CI/CD).
- **Phases**: Development → Continuous Integration → Continuous Testing → Continuous Deployment → Continuous Monitoring.
- **Best for**: Projects requiring rapid development cycles and ongoing maintenance.



10. Prototype Model

- **Approach**: Involves building a prototype (a working model of the software) early in the development process, which is then refined based on user feedback.
- **Phases**: Requirements → Prototype Design → Build Prototype → User Feedback → Refinement.
- **Best for**: Projects where user feedback is crucial in shaping the final product.

Each of these methodologies offers a different way to approach software development, and the choice of methodology depends on the project's size, complexity, and requirements.



Name = Pratik Kale

Batch = D_15