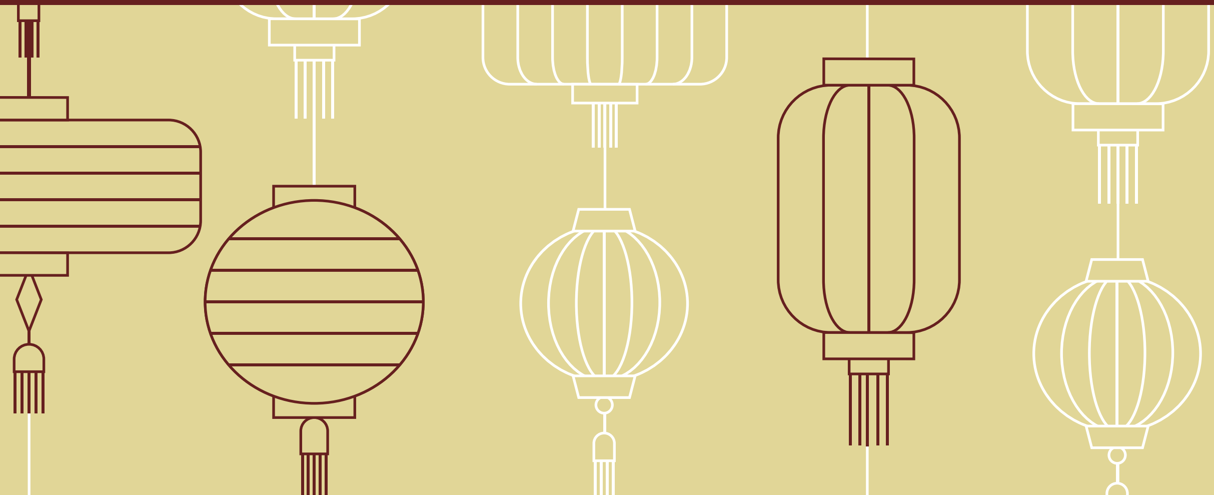
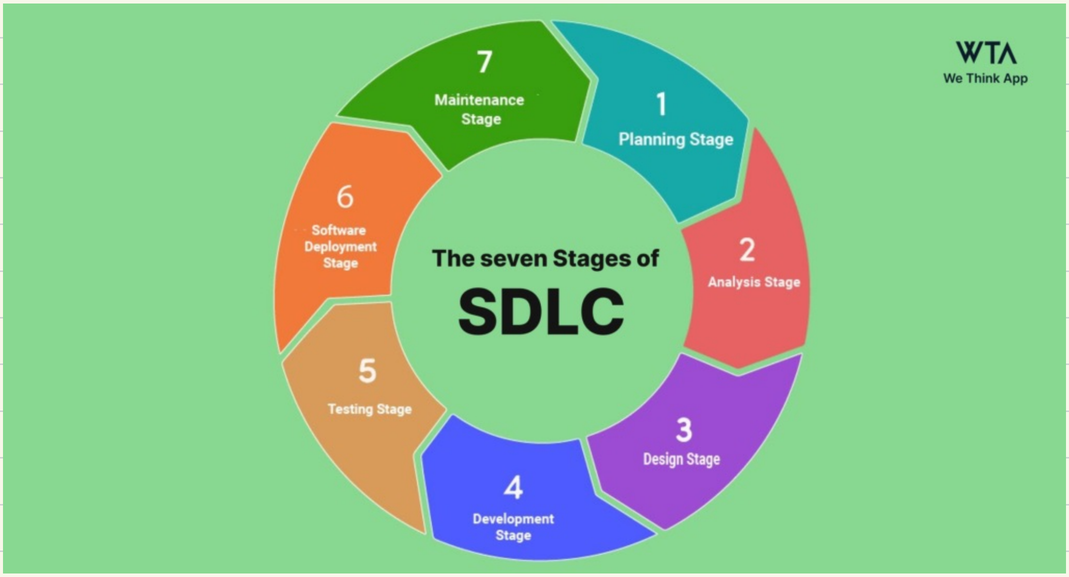


A. Introduction to Systems Development Life Cycle (SDLC)

The Systems Development Life Cycle (SDLC) is a structured approach to software development that outlines a series of stages through which software progresses from inception to deployment and maintenance.

The SDLC provides a framework for planning, creating, testing, and deploying an information system. It is designed to ensure that the development process is methodical, efficient, and effective in meeting the needs of the end-users and the organization.



Key Phases of SDLC:

- **Planning:** This initial phase involves understanding the project scope, determining the resources required, setting timelines, and identifying potential risks. It lays the foundation for the entire development process.

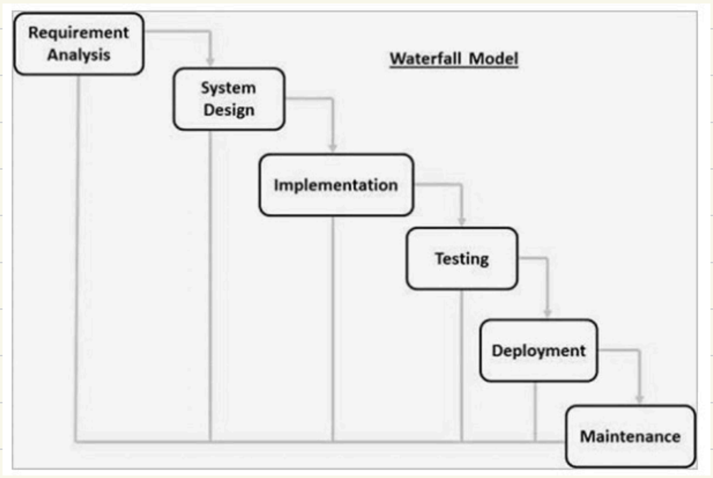
- **Requirements Analysis:** During this phase, detailed requirements of the system are gathered from stakeholders. This includes functional requirements (what the system should do) and non-functional requirements (how the system performs certain functions).

- **System Design:** This phase translates the requirements into a blueprint for constructing the system. It involves architectural design, interface design, data modeling, and detailed specifications.
- **Implementation (Coding):** In this phase, the actual code for the system is written based on the design specifications. It involves programming and integrating different components.
- **Testing:** This phase involves validating and verifying that the developed system meets all the specified requirements. Various testing methods are used to identify and fix bugs and ensure the system is reliable and performs as expected.
- **Deployment:** Once the system has been tested and is deemed ready for use, it is deployed to the production environment where end-users can start using it. This phase also includes user training and system documentation.
- **Maintenance:** After deployment, the system enters the maintenance phase, where it is monitored for performance issues, bugs are fixed, updates and enhancements are made, and support is provided to end-users.

B. SDLC Models

Waterfall Model

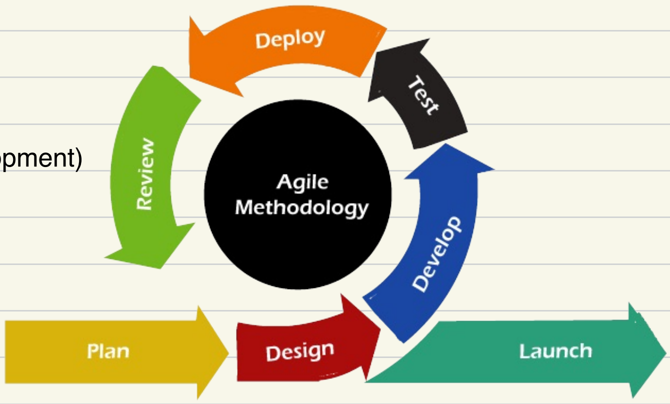
- Linear and sequential phases: Requirements, Design, Implementation, Testing, Deployment, Maintenance.
- **Advantages:** Clear structure, easy to manage, good for small projects with well-defined requirements.
- **Disadvantages:** Inflexibility to changes, late testing phase, not ideal for complex or long-term projects.



Agile Model

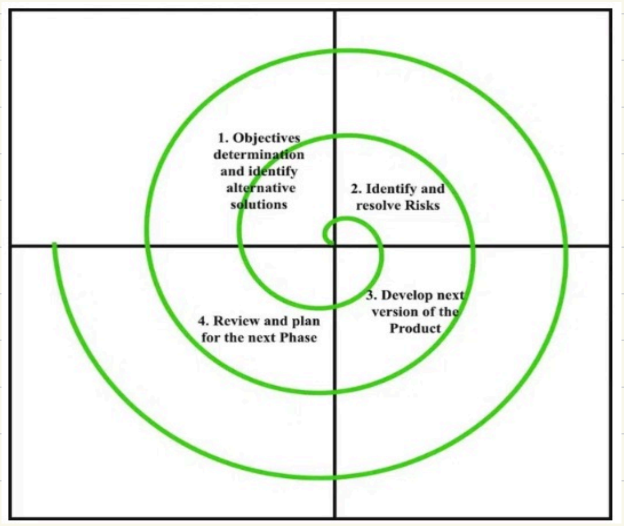
- Iterative and incremental development: Regular cycles of planning, development, testing, and review.
- **Advantages:** Flexibility, customer involvement, fast delivery of functional components, adaptability to changes.
- **Disadvantages:** Requires close collaboration, may lack documentation, challenging for large-scale projects without clear goals.

- The Agile Model includes popular methodologies like:
  - Scrum
  - Kanban
  - Extreme Programming (XP)
  - FDD (Feature-driven Development)



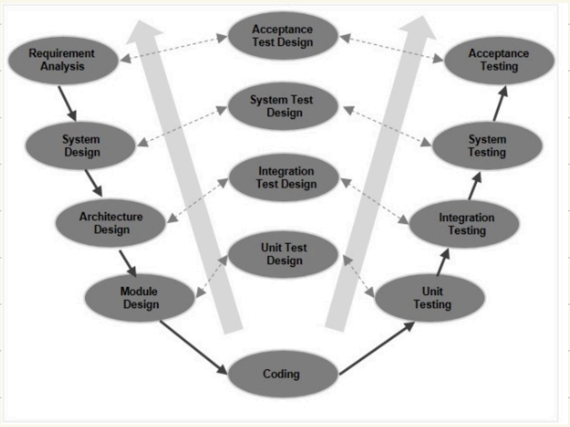
Spiral Model:

- Combines iterative development (prototyping) and systematic aspects of the waterfall model.
- **Advantages**: Risk management, iterative refinement, suitable for large and complex projects.
- **Disadvantages**: Can be costly, requires expertise in risk analysis, complex management.


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

V-Model (Verification and Validation):

- Extension of the Waterfall model with testing phases corresponding to each development phase.
- **Advantages**: Emphasizes verification and validation, structured approach, early detection of defects.
- **Disadvantages**: Rigid, not flexible to changes, similar drawbacks to the Waterfall model.



RAD Model (Rapid Application Development):

- Focuses on rapid prototyping and quick feedback over strict planning.
- **Advantages**: Speed, flexibility, customer feedback, useful for projects with well-understood requirements.
- **Disadvantages**: Requires highly skilled developers, may compromise quality for speed, not suitable for complex projects.

C. Comparing Traditional and Agile Approaches

1. **Project Management:**
   - Waterfall: Sequential, milestone-driven, heavy documentation, limited customer involvement.
   - Agile: Iterative, flexible, continuous customer feedback, adaptable planning.
1. **Flexibility:**
   - Waterfall: Low flexibility, changes are difficult to incorporate once the project progresses.
   - Agile: High flexibility, encourages changes even late in development.
2. **Outcomes:**
   - Waterfall: Predictable outcomes, but risks of misalignment with user needs if requirements change.

  - Agile: Potentially more aligned with user needs due to ongoing feedback, but outcomes may vary based on iterations.

D. Practical Applications of SDLC Models

1. **Successful Projects**:
    - Example: NASA's space shuttle software project using Waterfall due to its need for rigorous documentation and reliability.
    - Example: Spotify's use of Agile for its fast-paced development environment, enabling quick updates and user feedback integration.
2. **Failed Projects**:
    - Example: Denver Airport's baggage handling system using Waterfall, leading to delays and overspend due to inflexibility in requirements.
    - Example: HealthCare.gov's initial launch issues, where a lack of agile practices led to significant problems upon release.

E. SDLC Model Selection Criteria

- **Project Size and Complexity:**
    - Larger, more complex projects might benefit from models like Spiral or V-Model for their risk management and structured approach.
    - Smaller projects with clear requirements might be well-suited to Waterfall or RAD.
- **Risk and Uncertainty**:
    - Projects with high uncertainty or evolving requirements may benefit from Agile's iterative and flexible approach.
    - Projects with high risk and the need for thorough validation may benefit from the Spiral model's risk analysis focus.
- **Team Dynamics and Expertise**:
    - Agile requires teams with strong collaboration skills and experience with iterative development.
    - Traditional models like Waterfall may be easier for teams used to sequential, structured work processes.

- **Client Requirements and Involvement:**
    - Projects requiring continuous client involvement and feedback are ideal for Agile.
    - Projects with well-defined, unchanging requirements may be suitable for Waterfall or V-Model.

Learn More About SDLC in Action : https://www.geeksforgeeks.org/real-world-applications-of-sdlc-software-development-life-cycle/