SDLC Phases and Models

The SDLC is a structured process followed by software development teams to design, develop, test & and deploy software system.

Phases of SDLC

1. Requirement Analysis: It ~~inf~~ involves gathering and analyzing the requirements from stakeholders to understand what the Software should achieve.
2. Planning: It involves defining the project scope, resources, timeline, and budget. It includes creating a detailed project plan and risk management strategy.
3. Design: The architecture and design are created based on the requirements.
4. Implementation: Involves coding the software according to the design specifications. Developers write & compile the source code,
5. Testing: Involves verifying that the software works as intended and is free of bugs. This includes unit testing, integration testing, system testing, and user acceptance testing.
6. Deployment: It is the process of delivering the completed software to the production environment where it will be used by end-users.
7. Maintenance: It involves providing ongoing support to fix bugs, update the software, and improve performance base on user feedback.

Models

Waterfall Mode -> It follows a strict order: Requirement Analysis, Design, Implementation, Testing, Deployment, and Maintenance.

Pros: Simple & easy to understand
For fixed requirement projects
Easy to manage, for small projects

Cons: Inflexible to changes
Not for long term projects
Late discovery of issues.

V Model -> Verification and Validation model. It emphasizes parallel development and testing activities.

Pros: Each development phase has a corresponding testing phase.
Ensures early detection of defects.

Cons: Requires a lot of documentation
Difficult to adjust changes

Spiral Model -> It combines iterative development with waterfall & incremental. It focuses on risk management.

Pros: Risk management
Iterative for continuous improvement
for complex & high risk projects

Cons: Complex to manage
requires experts
High cost.

Agile Model → Iterative & incremental model .

It emphasizes on flexibility , customer collaboration , and rapid delivery of small , functional segments of the software .

Pros: Highly adaptable to changes

Continuous delivery

Close collaboration with stakeholders

Cons: Requires significant time & effort for frequent feedback .

Can be challenging to predict the effort required for each iteration .

Less documentation