Software Development Life Cycle (SDLC) Phases:

1. Planning
2. Requirement & Analysis
3. Design
4. Coding (development)
5. Implementation
6. Testing
7. Deployments
8. Review/Maintenance

SDLC is a structured, step-by-step approach for developing information systems.

SRS: Software Requirement Specifications.

Planning:

- Define the System to be developed.
- Set the project Scope.
- Develop the project Plan.

Requirement & Analysis:

- Gathering requirements (business requirements, RDD: Requirement Definition Document).
- Prioritizing requirements.

Design:

- Models, diagrams, GUIs, Screen Designs.

Development:

- **Coding:**
  - ✔ Implementing the design.
  - ✔ Make design.
- Database.
- Relational Tables.

Testing:

- Test conditions > perform test.
- Unit Testing (test individual unit of code).
- System Testing (check code works together).
- Integration Testing (working in another system).
- UAT (user Acceptance Testing) (checks user satisfaction).

Implementation phase:

- Write detailed user requirements.
- Provide Training to user.
  (Instruction manual for how to use the system)

Review/Maintenance:

- It ensures that the project should continuously meet the requirements.
- Providing Helpdesk and Support.

Decision of proper Implementation:

- **Parallel** (old and new system is compared).
- **Plunge** (jump on new).
- **Pilot** (testing system from small to large groups).
- **Phased** (implementation in phases).

SDLC Models:

1. WATERFALL Model:

Sequential based process.
Plan> requirement> design> development> implementation> testing> maintenance.
Feedback loop after every phase.
It's a Predictive model.

Assumptions:

- Requirement are fixed.

- Prior experience of members needed.
- Understanding requirements by all.
- Suited in low risk environment.

Advantages:

- Requirements stability
- Easy to understand milestones.
- Quality is important.

Disadvantages:

- Can't make changes.
- No preview happen.
- Going to previous phase is very costly (Eg: Requirement Phase takes approval from Planning phase then go to design phase).

2. **ITERATIVE Model:**

3. **V-MODEL (V: Variety):**

V: Verification and Validation

It's a Predictive Model (feedback does not allows change)

Testing takes too much time so V-model is introduced.

4. **SPIRAL Model:**

Risk involved.

It's an Adaptive Model.

Very good for Mission-critical projects.

Disadvantages:

- Not works well for smaller projects.
- Very Costly.
- High expertise.

5. **LEAN Model:**

Eliminate waste.

Work on time.

Focused to cut waste.

6. AGILE Model:

Fast.
Complete software preparation in 60-90 days.
Used for time critical methods.
AGILE is a mind-set.
It's an Adaptive Model.
Incremental and Iterative Model
Used widely.

7. PROTOTYPING Model:

a) Identifying basic requirements.
b) Develop initial prototype.
c) Review.
d) Enhance and revise.

Tools: JIRA

8. DEVOPS Model:

___Development.
___Operation.

Classifications:

1. Predictive V/S Adaptive.
2. Incremental V/S Iterative.

Predictive: Building on base of prediction, when requirement is fixed.
Adaptive: Build small then modifying it by feedback.
Incremental: Incrementally adding more things, done in phases.
Iterative: Replacing on existing model, cycle involved.