SS3 Computer Studies Notes

Topic: Software Development Life Cycle (SDLC) — Phases and Models

Meaning of Software Development Life Cycle (SDLC)

The Software Development Life Cycle (SDLC) is a step-by-step process used to plan, design, develop, test, install, and maintain software. Software is a set of programs and instructions that tells a computer what to do. Examples include Microsoft Word, school management systems, banking applications, and mobile apps.

Phases of the SDLC

Phase 1: Planning and Feasibility Study

The development team identifies the problem, sets the objectives, estimates the cost and time, and determines whether the project is possible.

Example: A school decides that it needs a system to record students' results.

Phase 2: Requirements Analysis: The team gathers information about what users need the software to do.

Example: Teachers need to enter scores, calculate totals, and print report cards.

Phase 3: System Design: The team plans how the software will look and work, including its interface, database, security, and features.

Example: Designing the student registration form and results database.

Phase 4: Development or Implementation: Programmers write the code using programming languages and development tools.

Example: Writing PHP or Python code to calculate students' examination scores.

Phase 5: Testing: The software is checked to identify errors and confirm that it meets the requirements.

Example: Entering sample scores to confirm that totals and grades are calculated correctly.

Phase 6: Deployment or Installation: The completed software is installed or made available to its users.

Example: Installing the school results system on the school's computers.

Phase 7: Maintenance: The software is updated, repaired, and improved after it has been deployed.

Example: Fixing a grading error or adding a new term to the school results system.

Summary of the phases

The phases can be remembered in this order:

Planning → Analysis → Design → Development → Testing → Deployment → Maintenance

Note: Some textbooks combine or divide these phases differently. Testing and development may also happen repeatedly in some SDLC models.

Meaning of an SDLC Model

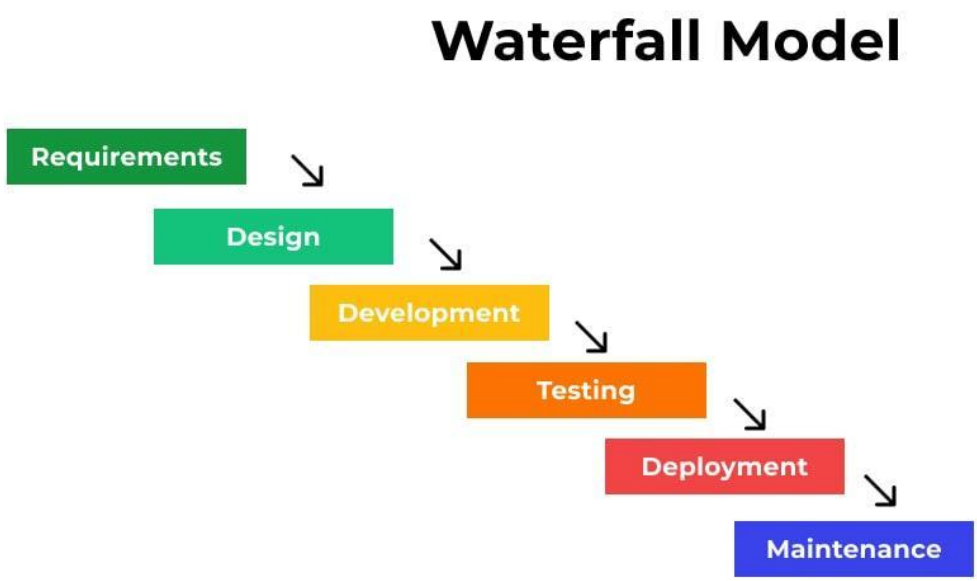
An SDLC model is an approach or method used to organise the phases of software development.

Different models arrange and manage the phases in different ways.

Two common models are:

1. Waterfall model
2. Agile model

Waterfall Model



The Waterfall model is a software development model in which work generally moves from one phase to the next in a fixed, sequential order.
The team normally completes one phase before moving to the next.

Characteristics of the Waterfall model

- It follows a planned sequence of stages.
- Requirements are usually defined early.
- Each phase is generally completed before the next begins.
- Documentation is important.
- Changes made later can be difficult and costly.

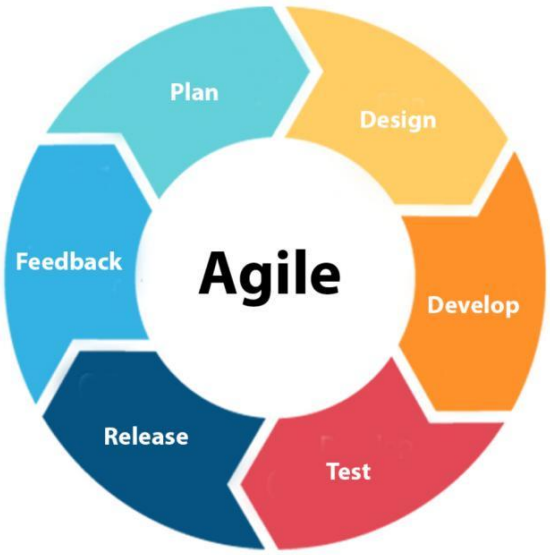
Advantages

1. It is simple to understand and manage.
2. The phases and responsibilities are clearly defined.
3. It is suitable for projects with stable requirements.
4. It is easier to estimate the time and cost when requirements are clear.

Disadvantages

1. It is difficult to accommodate major changes during development.
2. Users may not see a working version until late in the project.
3. Errors discovered late may be expensive to correct.
4. It may not suit projects whose requirements change frequently.

Agile Model



The Agile model is an approach to software development in which a team builds and improves software in small, repeated cycles, using regular feedback from users. Instead of waiting until the entire system is finished, the team delivers parts of the software and improves them over time.

Characteristics of the Agile model

- Development happens in small stages or iterations.
- Users provide feedback regularly.
- Testing takes place throughout development.
- Requirements can change as the project progresses.
- Teamwork and communication are important.

Advantages

1. It responds well to changing requirements.
2. Users can see working software earlier.
3. Feedback helps the team improve the product.
4. Problems can be identified during development.
5. Useful features can be delivered in stages.

Disadvantages

1. Frequent changes can make planning difficult.
2. Regular communication requires time and effort.
3. The total cost and completion date may be harder to predict.
4. Poor management can lead to repeated changes and delays.