


SDLC Phases and Models



 Software Engineering / SDLC
What is SDLC? Explain its phases.

Which SDLC model do you prefer and why?

Can you explain Agile methodology?

What are functional vs. non-functional requirements?

Sure—let's walk through each of these questions step by step:

1. What is SDLC? Explain its phases.

SDLC stands for **Software Development Life Cycle**.

It is a structured process that defines all the steps involved in developing software, from planning to maintenance.

Phases of SDLC (Traditional Waterfall example):

1. Requirement Gathering and Analysis

- Collect requirements from stakeholders.
- Analyze feasibility and document them.

2. System Design

- Define architecture, technologies, data models, and system interfaces.
- Create detailed design specifications.

3. Implementation (Coding)

- Developers write the actual code based on design documents.

4. Testing

- Testers verify the software to find defects.
- Perform unit testing, integration testing, system testing, and acceptance testing.

5. Deployment

- Release the software to the production environment for end-users.

6. Maintenance

- Fix bugs, make enhancements, and provide support as needed.

2. Which SDLC model do you prefer and why?

Answer Example (you can adjust to your context):

I prefer the **Agile model** because:

- It is **iterative and incremental**, delivering working software frequently.
- It allows **continuous feedback** from stakeholders, reducing the risk of building something that doesn't meet expectations.
- It quickly adapts to changing requirements.
- Testing is integrated throughout development, improving quality.

Other models you might mention (with a brief note):

- **Waterfall:** Good for small, well-understood projects.
- **V-Model:** Emphasizes verification and validation at each stage.
- **Spiral:** Focuses on risk analysis and prototyping.
- **Iterative:** Builds software in repeated cycles of development and refinement.

3. Can you explain Agile methodology?

Agile is an approach to software development that emphasizes:

- **Iterative development** (work in short cycles called sprints).
- **Collaboration** between cross-functional teams and customers.
- **Adaptability** to change rather than rigid plans.
- **Continuous delivery** of valuable software.

Key points:

- Work is divided into **sprints**, usually 2–4 weeks.
- At the end of each sprint, a potentially shippable product increment is delivered.
- Regular meetings are held:
 - **Daily stand-up:** Quick updates and blockers.
 - **Sprint planning:** Define what to build next.
 - **Sprint review:** Demonstrate the work done.
 - **Retrospective:** Reflect on what went well and what to improve.
- Common Agile frameworks:
 - **Scrum**
 - **Kanban**
 - **Extreme Programming (XP)**

4. What are functional vs. non-functional requirements?

Functional Requirements:

- Describe **what the system should do**.
- Define specific behavior or functions.
- Examples:
 - The system must allow users to log in.
 - When a payment is successful, send a confirmation email.

Non-Functional Requirements:

- Describe **how the system should be**.
- Define qualities or constraints of the system.
- Examples:
 - Performance: The system should handle 1,000 requests per second.
 - Usability: The interface must be user-friendly.
 - Security: Passwords must be encrypted.
 - Availability: 99.9% uptime.

If you'd like, I can help you prepare concise interview-ready answers or examples for each!



Give me answer in simple I was explain to interviewer. give me Interviewer style

Absolutely! Here's how you can **speak naturally to an interviewer**, in **simple language**, almost like you're explaining to them directly:

✓ 1 What is SDLC? Explain its phases.

"SDLC stands for Software Development Life Cycle. It's a process to build software step by step. It usually has these phases:

1. First, **Requirement Gathering**, where we talk to stakeholders to understand what they need.
2. Next, **Design**, where we plan the system architecture and how things will work.
3. Then, **Implementation**, which means actual coding.
4. After coding, we do **Testing**, to check for bugs and make sure it works as expected.
5. Once it's tested, we do **Deployment**, where we release the software to users.
6. Finally, there's **Maintenance**, where we fix issues and improve the system over time."

✓ 2 Which SDLC model do you prefer and why?

*"I prefer the **Agile model** because it's flexible. In Agile, we work in short cycles called sprints, and we deliver small parts of the software quickly. This way, we can get feedback early and make changes easily if*

requirements change. It helps reduce risks and keeps customers involved throughout the project."

✓ 3 Can you explain Agile methodology?

"Sure. Agile is a way of developing software in small parts instead of building everything at once. We divide the work into **sprints**, usually 2 to 4 weeks. After each sprint, we show what we built and get feedback. Agile focuses on teamwork, customer collaboration, and responding to change quickly. It includes practices like daily stand-up meetings, sprint planning, and retrospectives to continuously improve."

✓ 4 What are functional vs. non-functional requirements?

"Functional requirements are about **what the system should do**. For example, the system must let users register, login, or generate reports.

Non-functional requirements are about **how the system should be**. Like performance, security, and usability. For example, the website should load within 2 seconds or should be available 99% of the time."

If you want, I can help you polish these further or prepare more examples!