Lecture 6: Software Development Life Cycle (SDLC)

Understanding the Complete Development Process

State University of Zanzibar (SUZA)
BSc Computer Science

Contents

1 Introduction to SDLC

1.1 What is SDLC?

The Software Development Life Cycle (SDLC) is a systematic process for planning, creating, testing, and deploying software applications. It provides a structured framework that ensures:

- Quality software is delivered on time
- Development costs are controlled
- Customer requirements are met
- Risks are minimized

1.2 Why is SDLC Important?

1. **Predictability:** Provides clear milestones and deliverables
2. **Quality:** Ensures systematic testing and review
3. **Documentation:** Creates comprehensive project records
4. **Risk Management:** Identifies problems early
5. **Cost Control:** Prevents budget overruns

2 The Seven Phases of SDLC

2.1 Phase 1: Planning

Purpose: Define the project scope, objectives, and feasibility.

Key Activities:

- Define project goals and objectives
- Conduct feasibility study (technical, economic, operational)
- Identify stakeholders
- Create project timeline
- Allocate budget and resources
- Risk assessment

Deliverables:

- Project Charter
- Feasibility Study Report
- Project Plan

- Risk Assessment Document

Example Questions to Answer:

1. What problem are we solving?
2. Who are the users?
3. What is the budget and timeline?
4. Is this project technically feasible?

2.2 Phase 2: Requirements Analysis

Purpose: Gather and document detailed requirements.

Key Activities:

- Conduct stakeholder interviews
- Gather functional requirements (what the system does)
- Gather non-functional requirements (performance, security)
- Create user stories and use cases
- Prioritize requirements (MoSCoW method)
- Get stakeholder approval

Deliverables:

- Software Requirements Specification (SRS)
- User Stories
- Use Case Diagrams
- Requirements Traceability Matrix

MoSCoW Prioritization:

- **M**ust have - Critical requirements
- **S**hould have - Important but not critical
- **C**ould have - Nice to have
- **W**on't have - Out of scope for now

2.3 Phase 3: Design

Purpose: Create the blueprint for the software.

Key Activities:

- Design system architecture
- Create database schema
- Design user interface (UI/UX)
- Define API specifications
- Select technology stack
- Create component diagrams

Deliverables:

- System Design Document (SDD)
- Database Design (ERD)
- UI/UX Mockups and Wireframes
- API Documentation
- Architecture Diagrams

Types of Design:

1. **High-Level Design (HLD):** Overall system architecture
2. **Low-Level Design (LLD):** Detailed component design

2.4 Phase 4: Implementation (Coding)

Purpose: Write the actual code.

Key Activities:

- Set up development environment
- Write code following coding standards
- Implement features based on design
- Conduct code reviews
- Use version control (Git)
- Write unit tests

Best Practices:

- Follow coding conventions
- Write clean, readable code
- Comment complex logic
- Commit code frequently
- Review code before merging

2.5 Phase 5: Testing

Purpose: Verify the software works correctly.

Key Activities:

- Create test plans and test cases
- Perform unit testing
- Perform integration testing
- Perform system testing
- Conduct User Acceptance Testing (UAT)
- Fix bugs and retest

Types of Testing:

1. **Unit Testing:** Test individual components
2. **Integration Testing:** Test component interactions
3. **System Testing:** Test complete system
4. **UAT:** User validates requirements
5. **Performance Testing:** Test speed and scalability
6. **Security Testing:** Test for vulnerabilities

2.6 Phase 6: Deployment

Purpose: Release the software to production.

Key Activities:

- Prepare production environment
- Create deployment scripts
- Deploy application
- Configure servers and databases
- Perform smoke testing
- Train users
- Create user documentation

Deployment Strategies:

- **Big Bang:** Deploy everything at once
- **Rolling:** Gradual deployment to servers
- **Blue-Green:** Switch between two environments
- **Canary:** Deploy to small user group first

2.7 Phase 7: Maintenance

Purpose: Keep the software running and improve it.

Key Activities:

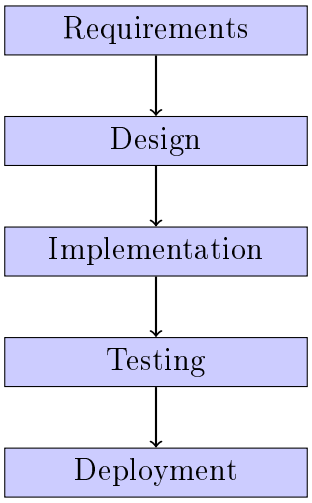
- Monitor system performance
- Fix bugs and issues
- Implement enhancements
- Apply security patches
- Optimize performance
- Update documentation

Types of Maintenance:

1. **Corrective:** Fixing bugs
2. **Adaptive:** Adapting to environment changes
3. **Perfective:** Adding new features
4. **Preventive:** Preventing future problems

3 SDLC Models

3.1 Waterfall Model



Characteristics:

- Sequential, linear approach
- Each phase must complete before next begins
- Well-documented

Best For: Projects with well-defined, stable requirements
Drawbacks: Inflexible, late testing, difficult to accommodate changes

3.2 Agile Model

Characteristics:

- Iterative and incremental
- Frequent deliveries (sprints)
- Embraces change
- Customer collaboration
- Working software over documentation

Popular Frameworks:

1. **Scrum:** Sprint-based with defined roles
2. **Kanban:** Visual workflow management
3. **XP:** Extreme Programming with pair programming

Best For: Projects with evolving requirements, need for flexibility

3.3 V-Model

Characteristics:

- Each development phase has corresponding testing phase
- Verification and validation at each stage
- Testing planned in parallel with development

Best For: Projects requiring high reliability (medical, aerospace)

3.4 Spiral Model

Characteristics:

- Risk-driven approach
- Combines iterative and waterfall
- Four phases repeated: Planning, Risk Analysis, Engineering, Evaluation

Best For: Large, complex, high-risk projects

4 Choosing the Right Model

Scenario	Recommended Model
Clear, fixed requirements	Waterfall
Evolving requirements	Agile
High-risk project	Spiral
Safety-critical system	V-Model
Quick prototype needed	Agile/Prototype

5 SDLC in Your Project

For your student projects, we recommend a **simplified Agile approach**:

1. **Week 1-2:** Planning and Requirements
2. **Week 3:** Design
3. **Week 4-6:** Sprint 1 (Core features)
4. **Week 7-9:** Sprint 2 (Additional features)
5. **Week 10-11:** Sprint 3 (Polish and testing)
6. **Week 12:** Deployment and Presentation

6 Summary

- SDLC provides a structured approach to software development
- Seven phases: Planning, Requirements, Design, Implementation, Testing, Deployment, Maintenance
- Different models suit different project types
- Agile is popular for modern software development
- Documentation is crucial throughout the process

7 Discussion Questions

1. What SDLC model would you choose for a mobile app with unclear requirements?
2. Why is the requirements phase so important?
3. What happens if you skip the design phase?
4. How does Agile handle changing requirements differently than Waterfall?