# **SDLC & STLC: A Complete Guide**

## **📖 Introduction**

SDLC (Software Development Life Cycle) and STLC (Software Testing Life Cycle) are structured approaches to building and testing software.

* SDLC focuses on development (planning, coding, deployment).
* STLC focuses on testing (verification, validation, defect tracking).

Both ensure high-quality, reliable software delivery.

## **🔄 SDLC (Software Development Life Cycle)**

### **Phases of SDLC**

| **Phase** | **Description** | **Key Activities** |
| --- | --- | --- |
| 1. Planning | Define project scope, goals, and resources. | Feasibility study, SRS document. |
| 2. Analysis | Gather detailed requirements. | Business & technical specs. |
| 3. Design | Create system architecture. | High-Level Design (HLD), Low-Level Design (LLD). |
| 4. Development | Write and build the software. | Coding, unit testing. |
| 5. Testing | Verify functionality (STLC runs here). | Integration, System, UAT. |
| 6. Deployment | Release to production. | CI/CD, rollout strategies. |
| 7. Maintenance | Fix bugs & improve performance. | Patches, updates. |

### **SDLC Models**

* Waterfall (Sequential)
* Agile (Iterative)
* Spiral (Risk-driven)
* DevOps (Continuous)

## **🔍 STLC (Software Testing Life Cycle)**

### **Phases of STLC**

| **Phase** | **Description** | **Key Activities** |
| --- | --- | --- |
| 1. Requirement Analysis | Understand testable requirements. | Identify test scenarios. |
| 2. Test Planning | Define strategy, scope, and tools. | Test plan document. |
| 3. Test Case Design | Create detailed test cases. | BVA, equivalence partitioning. |
| 4. Test Environment Setup | Prepare test data & infrastructure. | Mock APIs, test databases. |
| 5. Test Execution | Run test cases and log defects. | Manual/Automated testing. |
| 6. Test Cycle Closure | Evaluate testing effectiveness. | Test summary report. |

### **Testing Types in STLC**

✔ Functional Testing (Unit, Integration, System)  
✔ Non-Functional Testing (Performance, Security)  
✔ Regression Testing  
✔ User Acceptance Testing (UAT)

## **🆚 SDLC vs STLC**

| **Aspect** | **SDLC** | **STLC** |
| --- | --- | --- |
| Goal | Develop software. | Test software. |
| Phases | Planning → Maintenance. | Requirement Analysis → Test Closure. |
| Ownership | Developers, PMs, BAs. | QA Engineers, Testers. |
| Output | Functional software. | Test reports, bug logs. |
| Tools | JIRA, Git, Docker. | Selenium, TestRail, Postman. |

✅ SDLC & STLC work together – Testing starts early in SDLC (Shift-Left Testing).

## **🏆 Best Practices**

### **For SDLC**

✔ Clear documentation (SRS, HLD, LLD).  
✔ Code reviews for quality control.  
✔ CI/CD pipelines for faster releases.

### **For STLC**

✔ Early test planning (align with requirements).  
✔ Prioritize test cases (risk-based testing).  
✔ Automate regression tests (Selenium, Cypress).

### **For Both**

✔ Collaboration (Dev + QA teamwork).  
✔ Continuous feedback loops.  
✔ Monitor post-release (APM tools).

## **🌍 Real-World Examples**

### **Case 1: Healthcare.gov Crash (2013)**

* SDLC Failure: Rushed development, no load testing.
* STLC Failure: Skipped performance testing.
* Solution: Rebuilt with Agile SDLC + rigorous STLC.

### **Case 2: Facebook Outage (2021)**

* SDLC Issue: Faulty BGP update.
* STLC Gap: Lack of network failure testing.
* Fix: Added chaos engineering tests.

## **📚 References**

* [ISTQB SDLC & STLC Guidelines](https://www.istqb.org/)
* [Microsoft SDLC Docs](https://docs.microsoft.com/en-us/devops/plan/what-is-sdlc)
* [Agile Testing (Martin Fowler)](https://martinfowler.com/articles/agile-testing.html)

🔹 Conclusion:

* SDLC = Building software | STLC = Ensuring quality.
* Both are interdependent – strong SDLC needs strong STLC.
* Best practices reduce risks and improve efficiency.



