The software development life cycle (SDLC) is the process of planning, writing, modifying, and maintaining software.

**7 Stages of the System Development Life Cycle**

There are seven primary stages of the modern system development life cycle. Here’s a brief breakdown:

* Planning Stage
* Feasibility or Requirements of Analysis Stage
* Design and Prototyping Stage
* Software Development Stage
* Software Testing Stage
* Implementation and Integration
* Operations and Maintenance Stage

The systems development life cycle originally consisted of five stages instead of seven. These included planning, creating, developing, testing, and deploying. Note that it left out the major stages of analysis and maintenance.

The new seven phases of SDLC include planning, analysis, design, development, testing, implementation, and maintenance.

* Bug fixing – bugs are reported because of some scenarios that are not tested at all
* Upgrade – Upgrading the application to the newer versions of the Software.
* Enhancement – Adding some new features to the existing software.

### Phase 1: Requirement collection and analysis

The requirement is the first stage in the SDLC process. It is conducted by the senior team members with inputs from all the stakeholders and domain experts in the industry. Planning for the [quality assurance](https://www.guru99.com/all-about-quality-assurance.html) requirements and recognization of the risks involved is also done at this stage.

This stage gives a clearer picture of the scope of the entire project and the anticipated issues, opportunities, and directives which triggered the project.

Requirements Gathering stage need teams to get detailed and precise requirements. This helps companies to finalize the necessary timeline to finish the work of that system.

### Phase 2: Feasibility study

Once the requirement analysis phase is completed the next sdlc step is to define and document software needs. This process conducted with the help of ‘Software Requirement Specification’ document also known as ‘SRS’ document. It includes everything which should be designed and developed during the project life cycle.

**There are mainly five types of feasibilities checks:**

* **Economic:**Can we complete the project within the budget or not?
* **Legal:** Can we handle this project as cyber law and other regulatory framework/compliances.
* **Operation feasibility:** Can we create operations which is expected by the client?
* **Technical:** Need to check whether the current computer system can support the software
* **Schedule:** Decide that the project can be completed within the given schedule or not.

### Phase 3: Design

In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

This design phase serves as input for the next phase of the model.

There are two kinds of design documents developed in this phase:

**High-Level Design (HLD)**

* Brief description and name of each module
* An outline about the functionality of every module
* Interface relationship and dependencies between modules
* Database tables identified along with their key elements
* Complete architecture diagrams along with technology details

**Low-Level Design (LLD)**

* Functional logic of the modules
* Database tables, which include type and size
* Complete detail of the interface
* Addresses all types of dependency issues
* Listing of error messages
* Complete input and outputs for every module

### Phase 4: Coding

Once the system design phase is over, the next phase is coding. In this phase, developers start building the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process.

In this phase, Developer needs to follow certain predefined coding guidelines. They also need to use [programming tools](https://www.guru99.com/software-development-tools.html) like compiler, interpreters, debugger to generate and implement the code.

### Phase 5: Testing

Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement.

During this phase, QA and testing team may find some bugs/defects which they communicate to developers. The development team fixes the bug and send back to QA for a re-test. This process continues until the software is bug-free, stable, and working according to the business needs of that system.

### Phase 6: Installation/Deployment

Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

### Phase 7: Maintenance

Once the system is deployed, and customers start using the developed system, following 3 activities occur

* Bug fixing – bugs are reported because of some scenarios which are not tested at all
* Upgrade – Upgrading the application to the newer versions of the Software
* Enhancement – Adding some new features into the existing software

The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase.

## Popular SDLC Models

Here, are some of the most important models of Software Development Life Cycle (SDLC):

### [Waterfall model in SDLC](https://www.guru99.com/what-is-sdlc-or-waterfall-model.html)

The waterfall is a widely accepted SDLC model. In this approach, the whole process of the software development is divided into various phases of SDLC. In this SDLC model, the outcome of one phase acts as the input for the next phase.

This SDLC model is documentation-intensive, with earlier phases documenting what need be performed in the subsequent phases.

### [Incremental Model in SDLC](https://www.guru99.com/what-is-incremental-model-in-sdlc-advantages-disadvantages.html)

The incremental model is not a separate model. It is essentially a series of waterfall cycles. The requirements are divided into groups at the start of the project. For each group, the SDLC model is followed to develop software. The SDLC life cycle process is repeated, with each release adding more functionality until all requirements are met. In this method, every cycle act as the maintenance phase for the previous software release. Modification to the incremental model allows development cycles to overlap. After that subsequent cycle may begin before the previous cycle is complete.

### [V-Model in SDLC](https://www.guru99.com/v-model-software-testing.html)

In this type of SDLC model testing and the development, the phase is planned in parallel. So, there are verification phases of SDLC on the side and the validation phase on the other side. V-Model joins by Coding phase.

### [Agile Model in SDLC](https://www.guru99.com/agile-scrum-extreme-testing.html)

Agile methodology is a practice which promotes continue interaction of development and testing during the SDLC process of any project. In the Agile method, the entire project is divided into small incremental builds. All of these builds are provided in iterations, and each iteration lasts from one to three weeks.

### [Spiral Model](https://www.guru99.com/what-is-spiral-model-when-to-use-advantages-disadvantages.html)

The spiral model is a risk-driven process model. This SDLC testing model helps the team to adopt elements of one or more process models like a waterfall, incremental, waterfall, etc.

This model adopts the best features of the prototyping model and the waterfall model. The spiral methodology is a combination of rapid prototyping and concurrency in design and development activities.

### Big bang model

Big bang model is focusing on all types of resources in software development and coding, with no or very little planning. The requirements are understood and implemented when they come.

This model works best for small projects with smaller size development team which are working together. It is also useful for academic software development projects. It is an ideal model where requirements is either unknown or final release date is not given.