Applied Software Development Life cycle (SDLC)

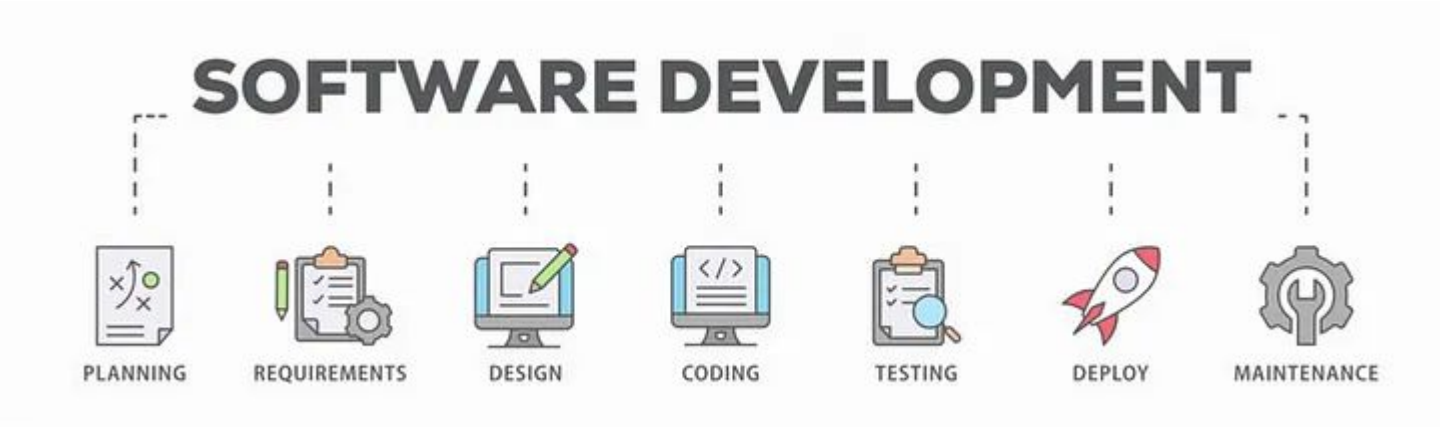
Anshuman Sanghvi,
Project Lead, Java.
STTL

Contents

- What is SDLC
- Phases of SDLC
- SDLC Models
- **SDLC applied** (we will mainly focus on this)

What is SDLC?

It is a framework you can apply to improve the predictability, efficiency and planning of your software development process. It is split into phases, with tasks and outcome defined for each phase.



Phases Of The SDLC

Types of SDLC Models

Waterfall

V Shape

Iterative

Spiral

Agile

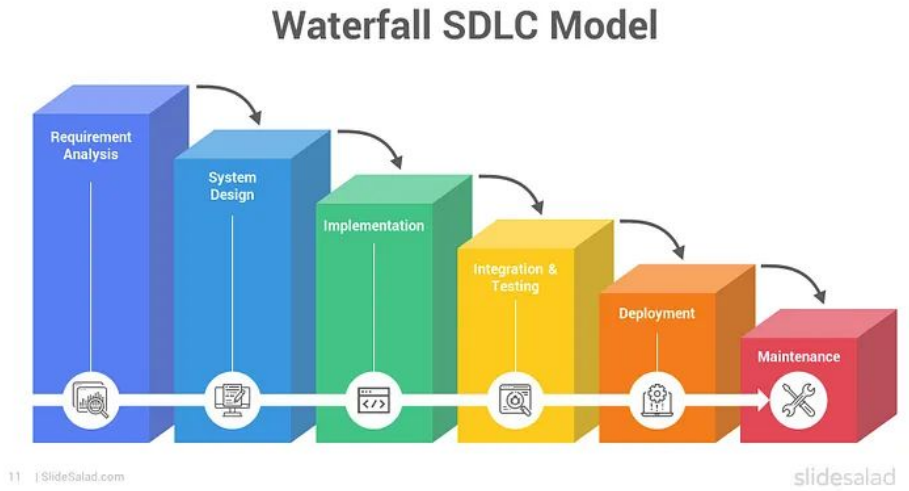
Types of SDLC Models

Waterfall

The waterfall model is the simplest and most straightforward among SDLC models.

It follows a linear and sequential approach, progressing through stages like project initiation, planning, requirement analysis, design, implementation, testing, and maintenance.

It's best suited for projects with clear, stable requirements and relatively short timelines, typically six months or less.



Waterfall

Advantages	Disadvantages
Simple to use and understand	The software is ready only after the last stage is over
Management simplicity thanks to its rigidity: every phase has a defined result and process review	High risks and uncertainty
Development stages go one by one	Not the best choice for complex and object-oriented projects
Perfect for the small or mid-sized projects where requirements are clear and not equivocal	Inappropriate for the long-term projects
Easy to determine the key points in the development cycle	The progress of the stage is hard to measure while it is still in the development
Easy to classify and prioritize tasks	Integration is done at the very end, which does not give the option of identifying the problem in advance

Types of Models

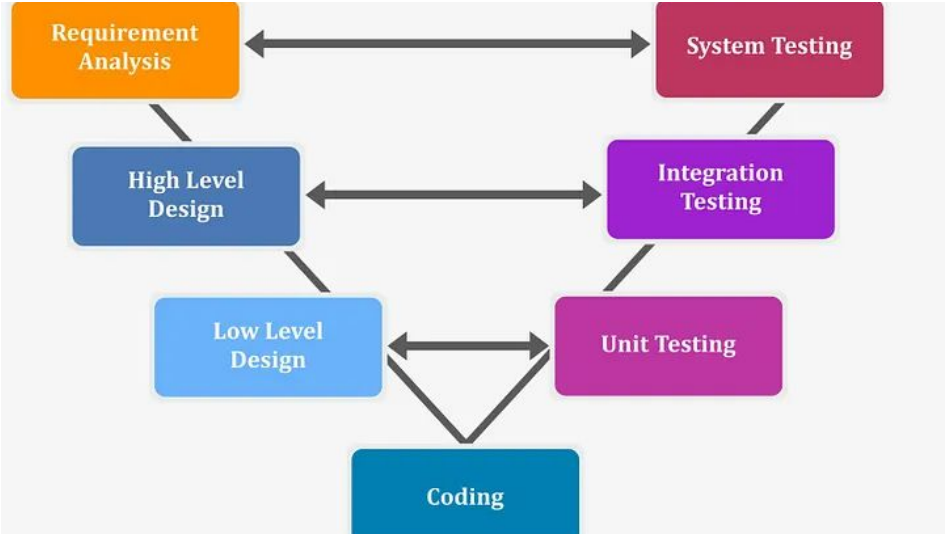
V Shape

For systems demanding high reliability, the V-shaped model is a go-to choice.

This model emphasizes rigorous testing, especially for critical systems where defects are not tolerable.

The process involves high-level design, detailed specifications, coding, unit testing, integration testing, and finally, operational testing.

The V-shape signifies the relationship between each development stage and its corresponding testing phase.



V Shape

Advantages	Disadvantages
Every stage of V-shaped model has strict results so it’s easy to control	Lack of the flexibility
Testing and verification take place in the early stages	Bad choice for the small projects
Good for the small projects, where requirements are static and clear	Relatively big risks

Types of Models

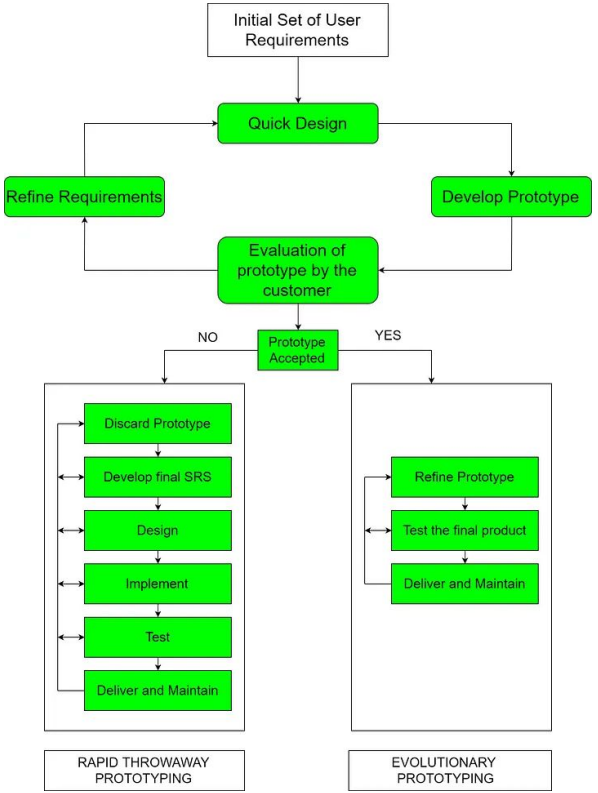
Iterative / Prototype

When there's a potential disconnect between customer understanding and the development team's grasp, the prototyping model steps in.

This approach involves creating visual representations or prototypes of the final product during the early stages.

This aids in clarifying requirements, ensuring a shared vision between the customer and the development team.

Prototyping proves particularly useful when customers find it challenging to visualize the end product based solely on specifications.



Iterative

Advantages	Disadvantages
Some functions can be quickly developed at the beginning of the development lifecycle	Iterative model requires more resources than the waterfall model
The paralleled development can be applied	Constant management is required
The progress is easy measurable	Issues with architecture or design may occur because not all the requirements are foreseen during the short planning stage
The shorter iteration is – the easier testing and debugging stages are	Bad choice for the small projects
It is easier to control the risks as high-risk tasks are completed first	The process is difficult to manage
Problems and risks defined within one iteration can be prevented in the next sprints	The risks may not be completely determined even at the final stage of the project

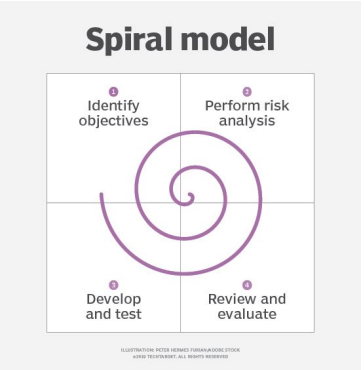
Types of Models

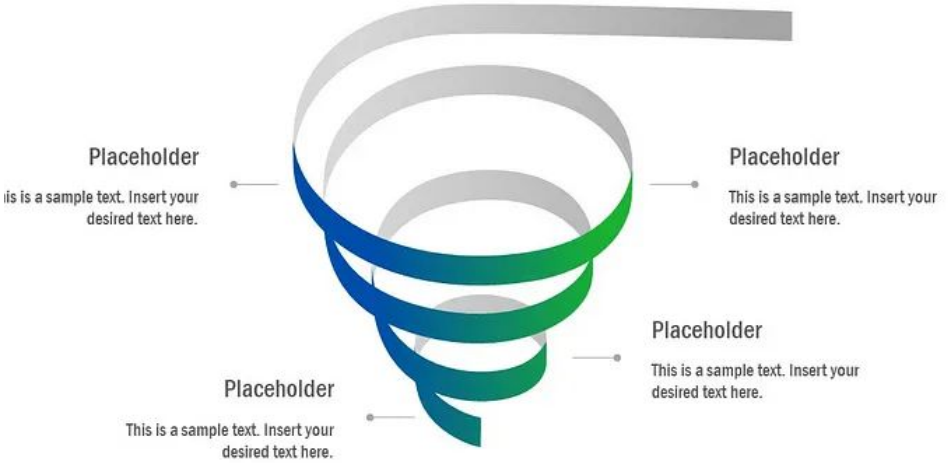
Spiral

Designed for projects with inherent risks, the spiral model combines elements of the iterative and prototyping models.

It involves cycles of planning, risk analysis, engineering, testing, and evaluation.

This iterative process helps manage and mitigate risks effectively, making it suitable for complex projects where uncertainties are prevalent.





Spiral

Advantages	Disadvantages
Lifecycle is divided into small parts, and if the risk concentration is higher, the phase can be finished earlier to address the treats	Can be quite expensive
The development process is precisely documented yet scalable to the changes	The risk control demands involvement of the highly-skilled professionals
The scalability allows to make changes and add new functionality even at the relatively late stages	Can be ineffective for the small projects
The earlier working prototype is done – sooner users can point out the flaws	Big number of the intermediate stages requires excessive documentation
Lifecycle is divided into small parts, and if the risk concentration is higher, the phase can be finished earlier to address the treats	Can be quite expensive
The development process is precisely documented yet scalable to the changes	The risk control demands involvement of the highly-skilled professionals

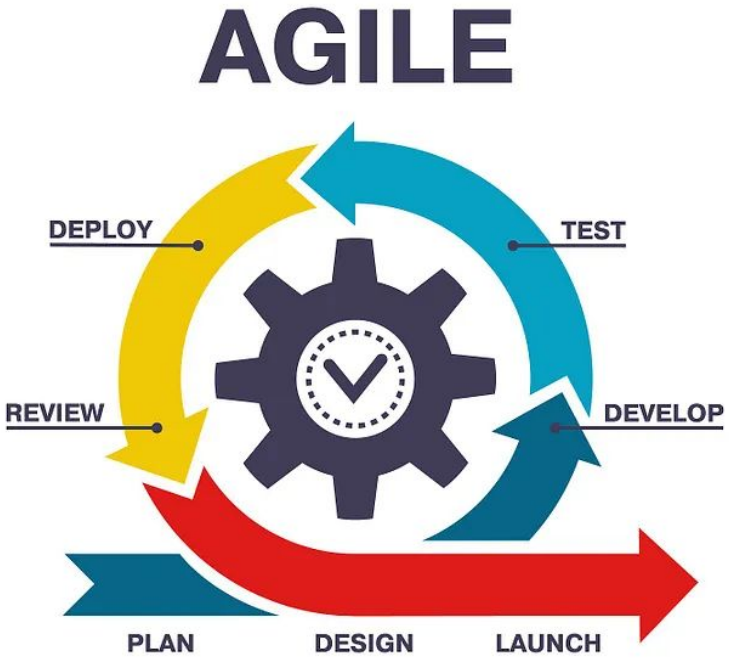
Types of Models

Agile

Agile methodologies, such as the Scrum model, have gained popularity for their adaptability to changing requirements.

Agile focuses on iterative development, collaboration, and customer feedback.

It’s particularly effective in dynamic environments where flexibility and responsiveness to change are paramount.



Agile

Advantages	Disadvantages
Corrections of functional requirements are implemented into the development process to provide the competitiveness	Difficulties with measuring the final cost because of permanent changes
Project is divided by short and transparent iterations	The team should be highly professional and client-oriented
Risks are minimized thanks to the flexible change process	New requirements may conflict with the existing architecture
Fast release of the first product version	With all the corrections and changes there is possibility that the project will exceed expected time

Comparison

Project Nature	Waterfall	V shape	Iterative	Spiral	Agile
High Risk & Unknowns				✅	
Incomplete Requirements			✅	✅	✅
High Correctness		✅			
High Complexity	✅	✅			
Frequent Output			✅		✅

Client RFP Statement: Develop a Software System that digitizes the attendance process for my company.

So let's actually apply the phases Of The SDLC

- As a simple example, we will look at RFC statement through each phase of the SDLC in a waterfall model.
- For each phase we will look at what it is, how we should work on it, why are we working on this phase, and what will be the expected outcomes of our work.
- Protip: If you focus on the **what** and **why**, you will have the clarity and motivation to learn the how and the outcome.

Requirement Analysis

What to do?
Identify stakeholders Understand their needs Translate needs into software requirements Check feasibility

Analysis: Features and Functional Requirements

How will I register in time and out time for an employee? What if the employee is working from different location?

Does the company have shifts? Are the shifts based on number of hours?

Are there provisions or consequences for coming late or leaving early?

What are the defined weekend off and holidays for the company? If company has branches in different states or countries, does the holiday list change per branch?

Are there roles outside a regular employee that should have different levels of access?

Analysis: Non Functional Requirements

How many active users will be using the application simultaneously?

Are there any security compliances or standards that need to be adhered to as per company policy?

Analysis: Scope and Feasibility

The company requires to use Optical Character Recognition on an Image that will analyze the signature of an employee when they sign a long term leave application and give response on whether the signature verifies the identity of the employee. Technically and Economically Feasible within given timeline and budget? Can you suggest alternatives?

The company requires approval for leave applications by an Employee. Scope: should you allow reversal of an approved leave? What is the maximum level of approvers to be supported?

Requirement Analysis

What to do?	How to do?	Why to do?	Expected Outcome
Identify stakeholders Understand their needs Translate needs into software requirements Check feasibility	Ask questions to client to known all the stakeholders (SH). Ask questions to SH and use domain knowledge to understand their needs. Use software systems knowledge to translate it into requirements. Check technical, hardware, knowledge, timeline feasibility of the requirements	To get clarity of: Who should the software be catered to What are its characteristics of the software What problems should it solve Whether it is possible to do so	Conclusion of whether you can and should do this development before you start development.

Planning

What to do?
Define scope of the software, activities required for it and estimation of resources for it: Define Functional and Nonfunctional Requirements, Use cases and user flows. Prepare milestones, priorities, dependencies, timelines.

Planning:

Does the client agree with the functional and nonfunctional requirements produced for software? Will there be a performance issue if we process the attendance of a month or year of all the employees at once?

What should be the tech stack required for this software? For e.g state machine for managing workflows. Are there any third party tools that require licenses? For e.g reporting tool or oracle database license. What should be the hardware requirements? How many application and database servers? Containerization? Master-Slave setup?

What components or features have a dependency on others? E.g marking attendance depends on definitions of shifts, weekends and holidays. Workflows depend on applications. Does that determine a development priority?

Planning:

How many resources are required and what is the KT required for finishing the software development in agreed time? Do I need to RnD for managing digital signatures?

What model of SDLC best suits this project? What should be other processes to be put in place? E.g Weekly planning and review meetings to track progress?

Are there any unknowns or risks that need to be clarified that affect the timeline, quality or outcome of the software?

Planning

What to do?	How to do?	Why to do?	Expected Outcome
Define scope of the software, activities required for it and estimation of resources for it: Define Functional and Nonfunctional Requirements, Use cases and user flows. Prepare milestones, priorities, dependencies, timelines.	Discuss with clients and stakeholders for agreeing requirements, priorities, approach and resources. Prepare list of manpower, technical and domain knowledge, technologies and hardware required. Include in the plan: configuration, initial set up, deployment, testing.	Validate understanding. Get mutual agreement of the task definition, approach, boundaries, risks, timelines, roles, responsibilities, costs involved and “definition of done” To avoid delays, errors, confusion, duplication, and miscommunication. To identify issues in advance, to prevent increment in scope.	RFP Response (deliverables, priorities, timelines, risks, SDLC Model, processes, resource utilization, budget utilization, tech stack) Software Requirement Specification Work Breakdown Structure

Design

What to do?
Based on the requirements, prepare a blueprint for what will be developed. Prepare High Level Design Prepare Low Level Design Prepare UI and UX design

Design:

Does the design or architecture diagram convey the non functional requirements or any other implied requirements?

Will the end users use the website from their phone? What should the UI/UX look like? Should it be mobile friendly?

Does this application also require a mobile app in the future? Do I need an API spec now?

What does the decision flow diagram of a leave application workflow look like? Should I allow rolling back the application? In that case I should add to the diagram.

What should be the database design? Does the design capture the relationships between objects? What should be the indices on the tables? How should I manage changes to the database design? Should I log the IP of the employee's login device that can help with monitoring?

Design

What to do?	How to do?	Why to do?	Expected Outcome
Based on the requirements, prepare a blueprint for what will be developed. Prepare High Level Design Prepare Low Level Design Prepare UI and UX design	Based on SRS: Prepare UI & UX wireframes, mockups and prototypes Prepare components' layers & interaction, user flow, data flow, sequence and database design diagrams.	Get the foundation of what you will build by turning abstract ideas into concrete details Clear communication of ideas and approach via architectural diagrams. Common reference for all stakeholders during all stages of SDLC.	Software Design Document. Agreed UI and UX. Agreed Software Architecture. Agreed Database Architecture.

Development

What to do?
Define coding meta work Convert Tasks into Coding tasks Code

Development:

Is there a requirement for a search engine or a cache engine or API gateway?

Are there any libraries required and do they have any compatibility issues with the existing setup?

Can the code readability and organization be enhanced using a software design pattern?

Does it make sense to develop the base of the feature as reusable code or library/framework?

Should the development of the feature consume an API? Should the task be performed via ORM or direct interaction with the database?

Does the software require support for multiple locales / timezones / languages?

Should this task be handled by a separate application or via a stored procedure or via a serverless function?

Development

What to do?	How to do?	Why to do?	Expected Outcome
Define coding meta work Convert Tasks into Coding tasks Code	Define development tools, CI/CD, code repositories & version control, coding styles, code quality and review practices. Analyze each task, translate into coding tasks, remove unknowns, prepare design, give estimate. Code according to Definition of Done (DoD).	Pre-agreed, well defined processes and standards for coding. Prevention of common coding issues.	Feature or Task done: According to the SRS, According to SDD, As per the DoD, Within estimated timeline, resources. Without causing regression. Project Source Code

Testing

What to do?
Prepare a test plan and process that ensures minimal bugs in the software. Prepare tools and technologies to be used for testing Test Validate as per SRS, SDD, DoD etc.

Testing:

What should be the testing unit and frequency? Which components require which kind of testing?

Prepare the test scenarios, and types of test required for each task.

Are there any compliances or standards that the application needs to be tested for?

Which tests should be automated?

What is the process for labelling the source, cause, affected surface area, importance and priority of a bug?

Testing

What to do?	How to do?	Why to do?	Expected Outcome
Prepare a test plan and process that ensures minimal bugs in the software. Prepare tools and technologies to be used for testing Test Validate as per SRS, SDD, DoD etc.	Analyze each task and plan for scope, timeline, priority and types of test (unit test, integration test, user acceptance test, stress test, regression test, security test) for each element of the software. Automate testing for suitable components.	Identify bugs To output software that is according to agreed acceptance criteria, quality and reliability standards.	Validated software according to SRS, SDD, DoD.

Deployment and Maintenance

What to do?
Prepare Software Release plan Prepare deployment details and plans. Deploy Prepare back up, roll back, monitoring, maintenance, and upgrade details and plans

Deployment and Maintenance:

What should be the process to change the schema of the database?

Which components and environments should have automated deployment via CI/CD and how many phases should they have?

How to log the application and manage the monitoring of the application?

How to manage environment variables and configurations? Should configuration and deployment information be maintained in a repository or documentation?

What are the data points to be monitored for this application?

Deployment and Maintenance

What to do?	How to do?	Why to do?	Expected Outcome
Prepare Software Release plan Prepare deployment details and plans. Deploy Prepare back up, roll back, monitoring, maintenance and upgrade details and plans	Prepare deployment order of components (front end, back end, middle ware, etc.), strategies, environments (developer, staging, and production), CI/CD pipelines, configuration, initialization, automation, backup strategy, roll back strategy as required. Post Deployment testing and monitoring System Upgrade and maintenance strategy	To make the product or its feature live. To prevent down time. To prepare mechanisms in advance that reduce any friction in the deployment or maintenance process.	Seamless deployment, maintenance, backup, roll back, and monitoring on any environment.

In summary: the slides so far hope to show the difference between

- a single person writing coding for a small application and

- software engineering in a large team for a large project

Example:

RFP Example (Show file)

SRS Example (Show file)

SDD Example (Show file)

Architecture Diagrams Example (Show file)

STTL PMS

Thank You

Skype: live:.cid.925bae56fe89963e
Email: anshuman . sanghvi @ silvertouch . com