Unit 1

Definition of Software Engineering, Software Development Life Cycle (SDLC) – Phases, Traditional vsAgile Models (Waterfall, Agile, DevOps), Scrum Basics – Roles, Sprint, Backlog, Version Control using Git and GitHub, Introduction to Project Tools (GitHub Projects, Jira, Trello)

1. SDLC Models

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

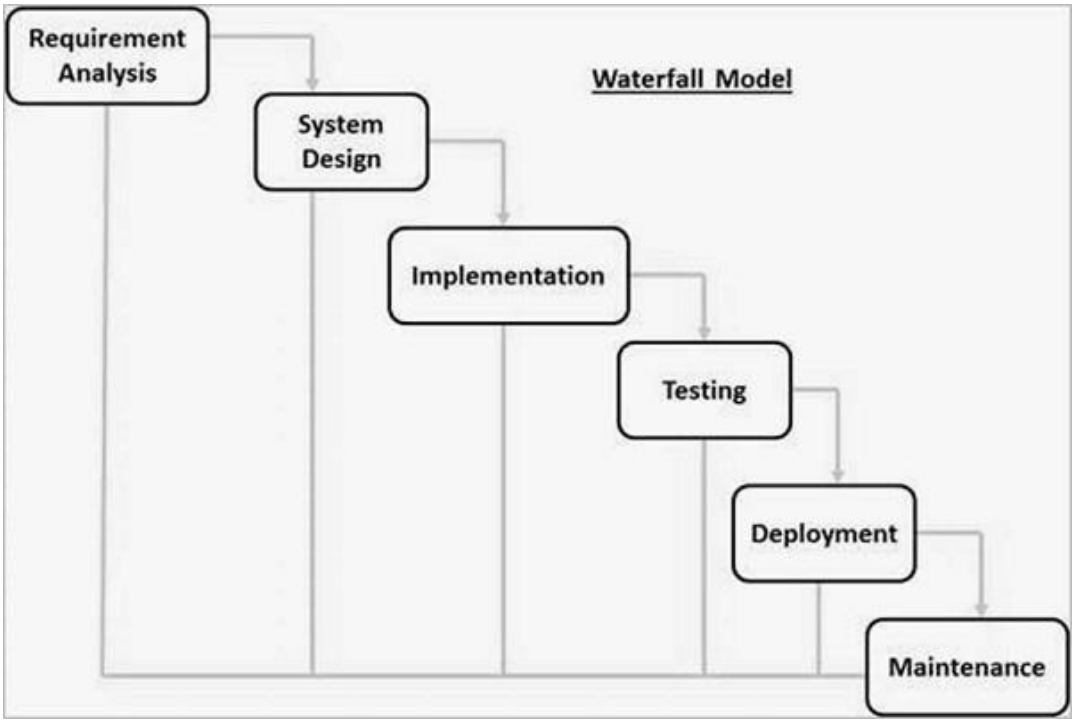
Following are the most important and popular SDLC models followed in the industry:

1. Waterfall Model
2. Prototyping
3. Iterative Model
4. V-Model
5. Spring Model
6. Agile process

1. Waterfall Model:

- Waterfall model is an example of a Sequential model. In this model, the software development activity is divided into different phases and each phase consists of series of tasks and has different objectives.
- It is divided into phases and output of one phase becomes the input of the next phase. It is mandatory for a phase to be completed before the next phase starts. In short, there is no overlapping in Waterfall model.
- In waterfall, development of one phase starts only when the previous phase is complete. Because of this nature, each phase of waterfall model is quite precise well defined. Since the phases fall from higher level to lower level, like a waterfall, It's named as waterfall model.

Pictorial representation of waterfall model:



Pros and Cons of waterfall model:

Advantages of using Waterfall model:

- Simple and easy to understand and use.
- For smaller projects, waterfall model works well and yield the appropriate results.
- Since the phases are rigid and precise, one phase is done one at a time, it is easy to maintain.
- The entry and exit criteria are well defined, so it easy and systematic to proceed with quality.
- Results are well documented.

Disadvantages of using Waterfall model:

- Cannot adopt the changes in requirements
- It becomes very difficult to move back to the phase. For example, if the application has now moved to the testing stage and there is a change in requirement, It becomes difficult to go back and change it.
- Delivery of the final product is late as there is no prototype which is

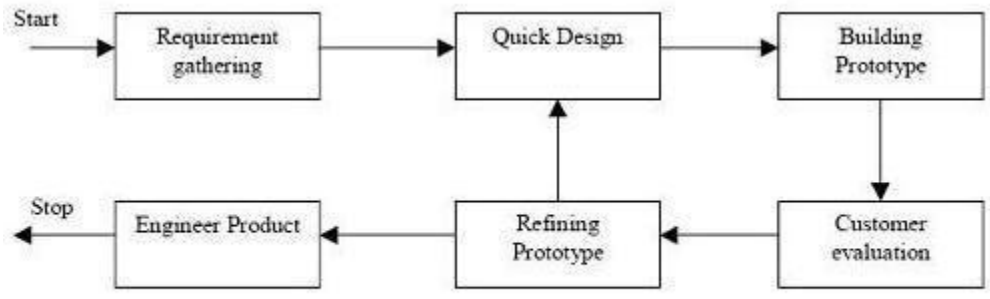
demonstrated intermediately.

- For bigger and complex projects, this model is not good as a risk factor is higher.
- Not suitable for the projects where requirements are changed frequently.
- Does not work for long and ongoing projects.
- Since the testing is done at a later stage, it does not allow identifying the challenges and risks in the earlier phase so the risk mitigation strategy is difficult to prepare.

2. Prototyping :

- The basic idea in Prototype model is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements.
- This prototype is developed based on the currently known requirements. Prototype model is a software development model. By using this prototype, the client can get an "actual feel" of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system.
- Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determining the requirements.
- The prototype are usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.

Diagram of Prototype model:



Prototyping Model

Advantages of Prototype model:

- Users are actively involved in the development
- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily

.

Disadvantages of Prototype model:

- Leads to implementing and then repairing way of building systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.

When to use Prototype model:

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model. It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
- Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system. They are excellent for designing good human computer interface systems.

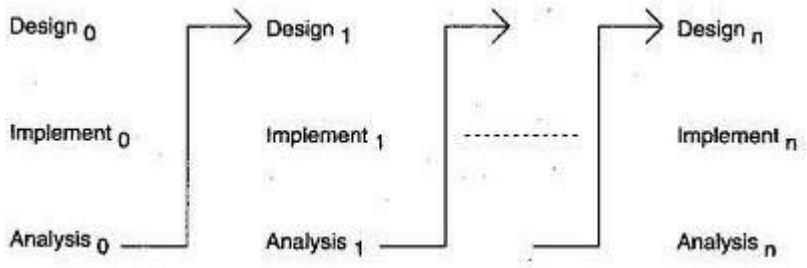
3. Iterative Model :

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.

For example:

In the diagram above when we work iteratively we create rough product or product piece in one iteration, then review it and improve it in next iteration and so on until it's finished. As shown in the image above, in the first iteration the whole painting is sketched roughly, then in the second iteration colors are filled and in the third iteration finishing is done. Hence, in iterative model the whole product is developed step by step.

Diagram of Iterative model:



Advantages of Iterative model:

- In iterative model we can only create a high-level design of the application before we actually begin to build the product and define the design solution for the entire product. Later on we can design and built a skeleton version of that, and then evolved the design based on what had been built.
- In iterative model we are building and improving the product step by step. Hence we can track the defects at early stages. This avoids the downward flow of the defects.
- In iterative model we can get the reliable user feedback. When presenting sketches and blueprints of the product to users for their feedback, we are effectively asking them to imagine how the product will work.
- In iterative model less time is spent on documenting and more time is given for designing.

Disadvantages of Iterative model:

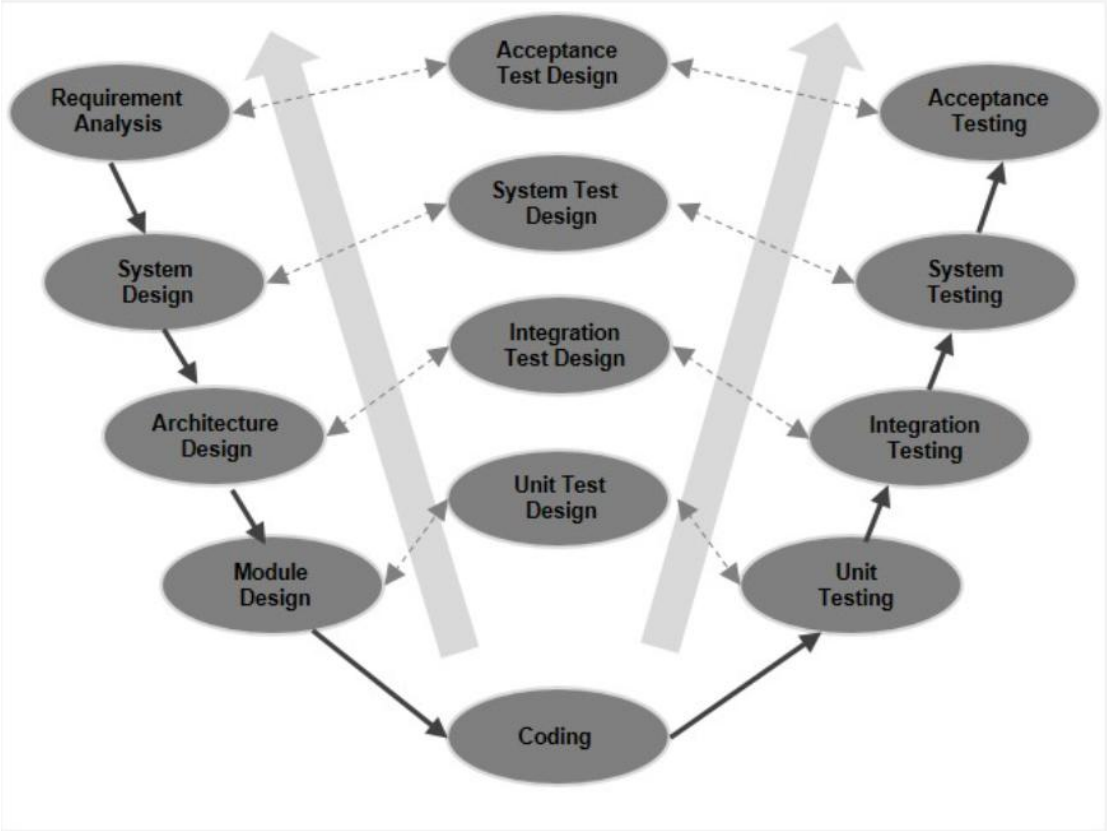
- Each phase of an iteration is rigid with no overlaps
- Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire lifecycle

When to use iterative model:

- Requirements of the complete system are clearly defined and understood.
- When the project is big.
- Major requirements must be defined; however, some details can evolve with time.

4.V-Model

- The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as Verification and Validation model.
- The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.
- In the V-Model, Verification phases on one side of the V and Validation phases on the other side. The Coding Phase joins the two sides of the V-Model.



V-Model - Verification Phases

There are several Verification phases in the V-Model,

Business Requirement Analysis

- This is the first phase in the development cycle where the product requirements are understood from the customers perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important activity and needs to be managed well, as most of the customers are not sure about what exactly they need. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.

System Design

- Once you have the clear and detailed product requirements, it is time to design the complete system. The system design will have the understanding and detailing the complete hardware and communication setup for the product under development. The system test plan is developed based on the system design.

Architectural Design

- Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. The system design is broken down further into modules taking up different functionality. This is also referred to as High Level Design (HLD).
- The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage.

Module Design

- In this phase, the detailed internal design for all the system modules is specified, referred to as Low Level Design (LLD). It is important that the design is compatible with the other modules in the system architecture and the other external systems.
- The unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. These unit tests can be designed at this stage based on the internal module designs.

Coding Phase

- The actual coding of the system modules designed in the design phase is taken up in the Coding phase. The best suitable programming language is decided based on the system and architectural requirements.
- The coding is performed based on the coding guidelines and standards. The code goes through numerous code reviews and is optimized for best performance before the final build is checked into the repository.

Validation Phases

The different Validation Phases in a V-Model

Unit Testing

- Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.

Integration Testing

- Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.

System Testing

- System testing is directly associated with the system design phase. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during this system test execution.

Acceptance Testing

- Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non-functional issues such as load and performance defects in the actual user environment.

Advantages of V-model:

- This is a highly-disciplined model and Phases are completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Simple and easy to understand and use.
- Each phase has specific deliverables and a review process.

Disadvantages of V-model:

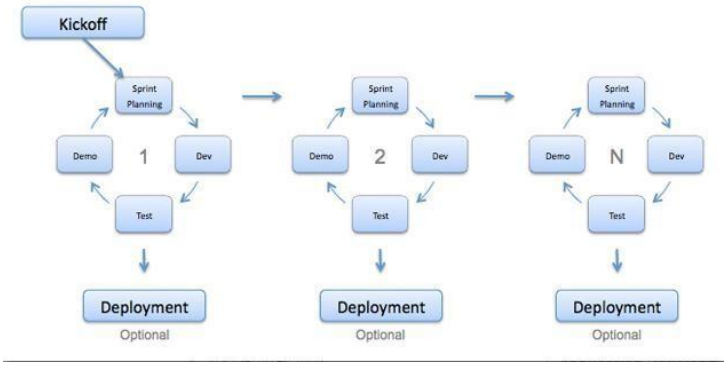
- High risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.

- Not suitable for the projects where requirements are at a moderate to high risk of changing.
- No working software is produced until late during the life cycle**.**

5.Agile process Model:

Agile development model is also a type of Incremental model. Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained. It is used for time critical applications. Extreme Programming (XP) is currently one of the most well known agile development life cycle model.

Diagram of Agile model:



Advantages of Agile model:

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

Disadvantages of Agile model:

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

When to use Agile model:

- When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
- Unlike the waterfall model in agile model very limited planning is required to get started with the project. Agile assumes that the end users' needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
- Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

Differences between Traditional and Agile Model

- Traditional project management follows a sequential, plan-driven approach with rigid processes, whereas agile embraces an iterative, flexible methodology.
- In traditional methods, there is extensive upfront planning, however, agile adapts to changes through continuous feedback and improvement cycles.
- Furthermore, traditional projects have clear roles, responsibilities, and long approval processes, unlike Agile's self-organizing, cross-functional teams that facilitate frequent deliveries.
- Consequently, traditional suits projects with stable requirements, while on the other hand, agile excels in dynamic environments with evolving needs.
- Ultimately, traditional methods excel in predictability, while in contrast, agile thrives on responsiveness to change.

Traditional vs Agile Models (Waterfall, Agile, DevOps)

Traditional Software Development	Agile Software Development
It is used to develop simple software.	It is used to develop complicated software.
In this methodology, testing is done once the development phase is completed.	In this methodology, testing and development processes are performed concurrently.
It follows a linear organizational expectation structure.	It follows an iterative organizational structure.
It provides less security.	It provides high security.
Client involvement is less as compared to Agile development.	Client involvement is high as compared to traditional software development.
It provides less functionality in the software.	It provides all the functionality needed by the users.
It supports a fixed development model.	It supports a changeable development

	model.
It is used by freshers.	It is used by professionals.
Development cost is less using this methodology.	Development cost is high using this methodology.
It majorly consists of five phases.	It consists of only three phases.
It is less used by software development firms.	It is normally used by software development firms.
The expectation is favored in the traditional model.	Adaptability is favored in the agile methodology.
Traditional software development approaches are formal in terms of communication with customers.	Agile software development methodologies are casual. In other words, customers who work with companies that utilize Agile software development approaches are more likely to interact with them than customers who work with companies that use traditional software development methodology.
For starters, typical software development approaches employ a predictive approach. There is full specification and prediction of the software development processes because the product is produced through rigorous and explicit planning. Changes are not permitted in this technique because the time and cost of project development are fixed.	Here, a flexible approach is used as the software development approaches are founded on the notion of continual design improvement and testing relies on team and client feedback.
Examples • Office productivity suites • Data management software • Media players • Security programs	Examples • Sky • Phillips • JP Morgan Chase

Agile	**DevOps**
Primarily on software development	**On end-to-end software delivery, from development to operations**
Iterative development, sprints, daily stand-ups	**Continuous integration, continuous delivery, automated testing**
Within development teams and with customers	**Across all IT departments, including development and operations**
Short feedback cycles within sprints	**Continuous feedback throughout the software lifecycle**
Rapid product evolution based on user feedback and changing needs	**Faster and more reliable product releases**
Emphasizes flexibility and customer collaboration	**Focuses on breaking divisions and fostering communication across teams**