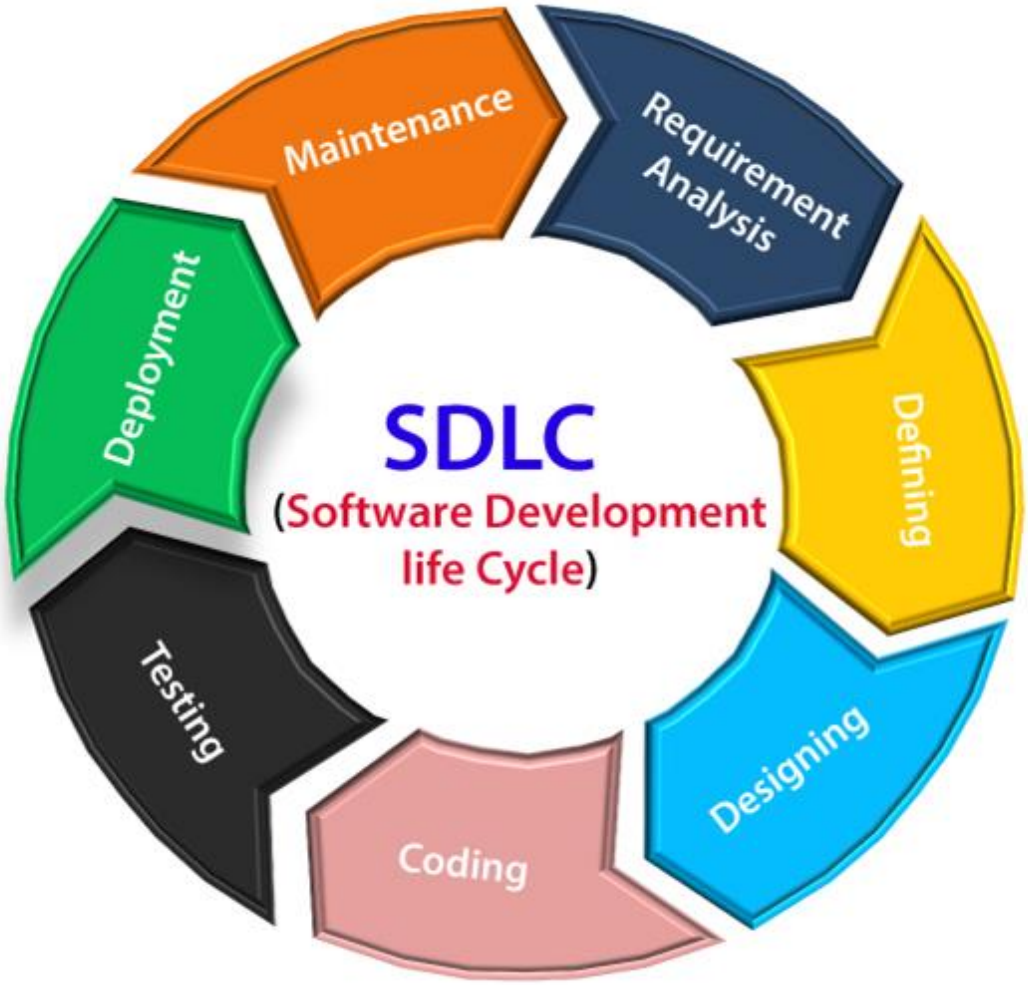
Software Development Life Cycle (SDLC)

The Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop, and test high-quality software. It aims to produce software that meets or exceeds c ustomer expectations, reaches completion within times and cost esti mates, and is reliable and maintainable.

SDLC Cycle represents the process of developing software. SDLC framework includes the following steps:



Phases of SDLC

1. **Planning**:
    - **Purpose**: Define the project scope, objecti ves, and resources. Identify risks and creat e a project schedule.
2. **Requirement Analysis**:
    - **Purpose**: Gather and analyze the require ments from stakeholders. Create a detaile d requirements specification document.
3. **Design**:
    - **Purpose**: Create the architectural designs, define system architecture, and specify sy stem modules and data flow.
4. **Implementation (Coding)**:
    - **Purpose**: Convert design documentation i nto the actual code.
5. **Testing**:
    - **Purpose**: Detect and fix defects, verify tha t the software meets the specified require ments.

6. **Deployment**:
    - **Purpose**: Release the software to a production environment.
7. **Maintenance**:
    - **Purpose**: Provide ongoing support and updates to the software to fix issues, improve functionality, and ensure compatibility with evolving technologies

What is unit and integration testing?

Unit Testing

- **Definition**: Unit testing is the process of testing individual units or components of the software to ensure that each unit functions as expected.
- **Purpose**: Verify that each unit of the software performs as designed.
- **Tools**: JUnit, NUnit, MSTest.

Integration Testing

- **Definition:** Integration testing involves combining individual units and testing them as a group to ide ntify any issues in the interactions between integra ted units.
- **Purpose:** Ensure that combined units function tog ether correctly.
- **Types:**
  - **TopDown Integration:** Testing starts from the top of the control flow and progresses downward.
  - **BottomUp Integration:** Testing starts from the bottom of the control flow and progresses upward.
- **Tools**: Selenium, JUnit, TestNG.

What is Risk Management?

Risk Management

A software project can be concerned with a large variety of risks. In order to be adept to systematically identify the significant risks which might affect a software project, it is essential to classify risks into different classes. The project manager can then check which risks from each class are relevant to the project.

There are three main classifications of risks which can affect a software project:

1. Project risks
2. Technical risks
3. Business risks

1. Project risks: Project risks concern differ forms of budgetary, schedule, personnel, resource, and customer-related problems. A vital project risk is schedule slippage. Since the software is intangible, it is very tough to monitor and control a software project. It is very tough to control something which cannot be identified. For any manufacturing program, such as the manufacturing of cars, the plan executive can recognize the product taking shape.

2. Technical risks: Technical risks concern potential method, implementation, interfacing, testing, and maintenance issue. It also consists of an ambiguous specification, incomplete specification, changing specification, technical uncertainty, and technical obsolescence. Most technical risks appear due to the development team's insufficient knowledge about the project.

3. Business risks: This type of risks contain risks of building an excellent product that no one need, losing budgetary or personnel commitments, etc.

Other risk categories

1. **1. Known risks:** Those risks that can be uncovered after careful assessment of the project program, the business and technical environment in which the plan is being developed, and more reliable data sources (e.g., unrealistic delivery date)
2. **2. Predictable risks:** Those risks that are hypothesized from previous project experience (e.g., past turnover)
3. **3. Unpredictable risks:** Those risks that can and do occur, but are extremely tough to identify in advance.

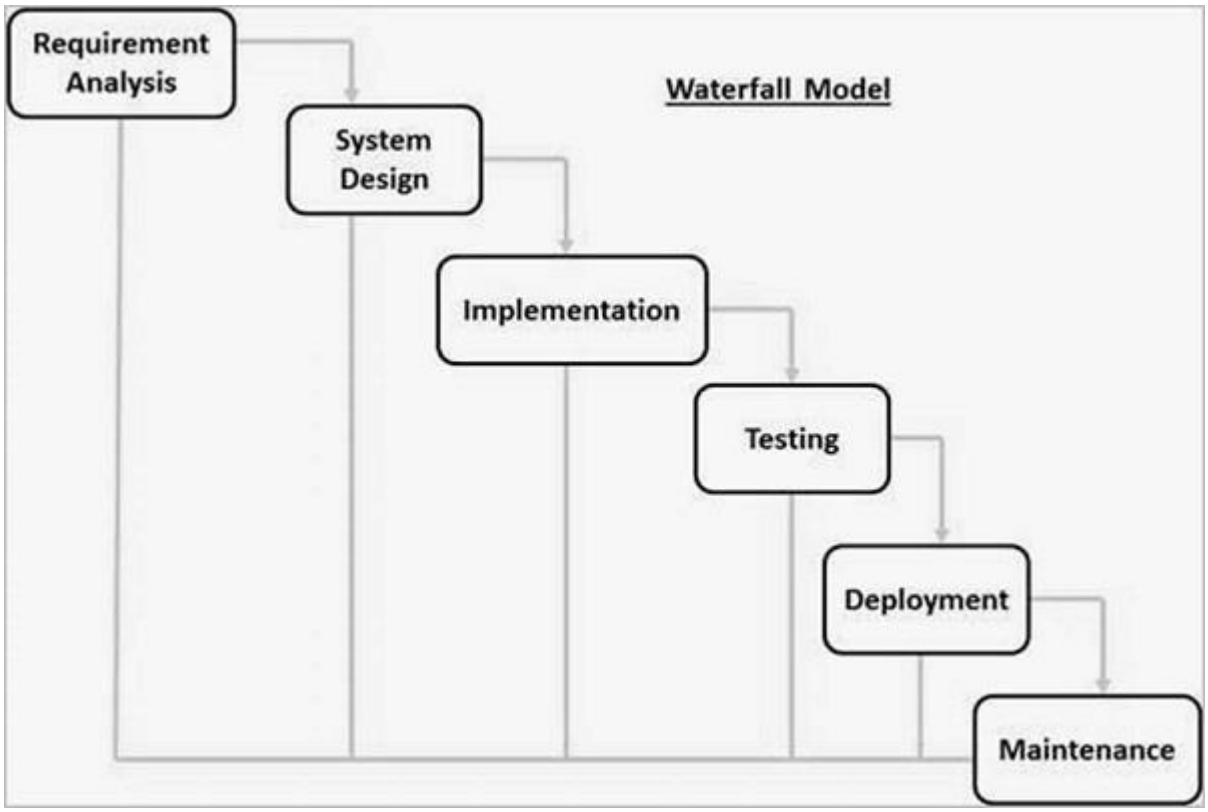
Principle of Risk Management

1. **Global Perspective:** In this, we review the bigger system description, design, and implementation. We look at the chance and the impact the risk is going to have.
2. **Take a forward-looking view:** Consider the threat which may appear in the future and create future plans for directing the next events.
3. **Open Communication:** This is to allow the free flow of communications between the client and the team members so that they have certainty about the risks.
4. **Integrated management:** In this method risk management is made an integral part of project management.
5. **Continuous process:** In this phase, the risks are tracked continuously throughout the risk management paradigm.

Waterfall Model - Design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.



The sequential phases in Waterfall model are –

- **Requirement Gathering and analysis** – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Waterfall Model - Advantages

- *Simple and easy to understand and use*
- *Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.*
- *Phases are processed and completed one at a time.*
- *Works well for smaller projects where requirements are very well understood.*

Waterfall Model – DisAdvantages

- *No working software is produced until late during the life cycle.*
- *High amounts of risk and uncertainty.*
- *Not a good model for complex and object-oriented projects.*
- *Poor model for long and ongoing projects.*

Application of Waterfall Model

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

What is project planning?

The project planning phase of the Software Development Life Cycle (SDLC) is a core stage that establishes the foundation for the project's success. It involves a number of activities, including:

- **Defining objectives**: The project's purpose, goals, and desired results are identified.
- **Building a team**: The development team is assembled.
- **Gathering requirements**: Requirements are collected from stakeholders, such as customers, managers, and experts.
- **Defining the workflow**: The project's workflow is defined.

- **Conducting a feasibility study**: A feasibility study and risk assessment is performed.
- **Choosing a tech stack**: The right tech stack is selected.
- **Analyzing resources and costs**: The resources and costs needed to complete the project are analyzed.
- **Setting deadlines**: Deadlines and time frames are set for each phase of the SDLC.
- **Assigning roles**: Roles are assigned to the development and analysis teams.
- **Planning a budget**: A budget is planned for the project.
- **Providing an overview of customer expectations**: An overview of customer expectations is provided.