**SDLC – SOFTWARE DEVELOPMENT LIFE CYCLE**

The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software.

Phases of software development:

* Requirement analysis
* Design
* Coding
* Testing
* Deployment
* Maintenance

In Requirement Analysis phase,

1. Functional Requirements
2. Non-Functional Requirements

**Functional Requirements:**

These are the requirements that the end user specifically demands as basic facilities that the system should offer. A functional requirement defines what a system must or must not, do.

**Non-Functional Requirements:**

These are basically the quality constraints that the system must satisfy according to the project contract.

They basically deal with issues like:

* Portability
* Security
* Maintainability
* Reliability
* Scalability
* Performance
* Reusability
* Flexibility

In Design phase,

**High-Level Design (HLD) and Low-Level Design (LLD):**

* High-Level Design:

HLD is the general system design means it refers to the overall system design. The HLD involves system architecture, database design, a brief description of systems, services, platforms, and relationships among modules. The HLD is also known as macro-level or system design. It changes the business or client requirement into a High-Level Solution.

Example: Our college website.

* Low-Level Design:

The LLD stands for Low-Level Design, in which the designer will focus on the components like a User interface (UI). The Low-level design is created by the developer manager and designers. It is also known as micro-level or detailed design. The LLD can change the High-Level Solution into a detailed solution.

Example: Each module of our college website.

**SDLC Methodologies:**

There are many types of SDLC models of which some includes,

* Waterfall Model
* RAD Model
* Spiral Model
* V- Model
* Incremental Model
* Agile Model
* Iterative Model
* Bigbang Model

**Advantages and Disadvantages of Models:**

**Waterfall Model:**

**Advantages:**

* Before the next phase of development, each phase must be completed.
* Suited for smaller projects where requirements are well defined.
* Elaborate documentation is done at every phase of the software’s development cycle.

**Disadvantages:**

* Error can be fixed only during the phase.
* It is not desirable for complex project where requirement changes frequently.
* Testing period comes quite late in the developmental process.
* Clients valuable feedback cannot be included with ongoing development phase.

**RAD Model:**

**Advantages:**

* It is useful when you have to reduce the overall project risk.
* It is adaptable and flexible to changes.
* It is easier to transfer deliverables as scripts, high-level abstractions and intermediate codes are used.
* Due to code generators and code reuse, there is a reduction of manual coding.

**Disadvantages:**

* Not all application is compatible with RAD.
* It can’t be used for smaller projects.
* When technical risk is high, it is not suitable.
* If developers are not committed to delivering software on time, RAD projects can fail.

**V-Model:**

**Advantages:**

* Higher success chances because development of test plans early on during the life cycle.
* Verification and validation of the product in the early stages of product development.
* An apt fit for small projects with easily understandable requirements and easy to use.

**Disadvantages:**

* Costly and required more time, in addition to a detailed plan
* Less flexible like waterfall model.
* Requirement and test documents need to be updated if any changes had to be made amid the software development.

**Spiral Model:**

**Advantages:**

* Early involvement of developers.
* Manages risks and develops the system into phases.
* As the prototype build is done in small increments, cost estimation is easy.

**Disadvantages:**

* Much more documentation due to intermediate phases
* High cost and time to reach the final product.
* Needs special skills to evaluate the risks and assumptions.
* Highly customized limiting re-usability.

**Agile Model:**

**Advantages:**

* Higher success chances because development of test plans early on during the life cycle.
* Verification and validation of the product in the early stages of product development.
* An apt fit for small projects with easily understandable requirements and easy to use.

**Disadvantages:**