Module 1: Software Development Process

- Software engineering vs software development
- The SDLC
- SDLC Models advantages and disadvantages

Module 1: Software Development Process

I. Software engineering vs software development

Software engineering is the application of engineering principles to the design, implementation, deployment and maintenance of software systems

Software development are the various activities carried out in order to successfully conception design and implementation of a software system

As software engineers, we promote the use of a scientific approach to the process of software development.

Why Scientific Approach?

The choice of adopting a scientific approach to software development is due to a period known as the software crisis

The term "software crisis" refers to the challenges faced in software development during the 1960s and 1970s. It arose from the struggle to keep up with the rapid increase in:

- **Software Demand:** The need for software grew exponentially as computers became more powerful and accessible.
- **Software Complexity:** As software tackled bigger problems, it became more intricate and difficult to manage.

It became a crisis due to the following reasons:

- **Missed Deadlines:** it was difficult for software engineers to correctly estimate the duration of their projects and deliver on time
- **Budget Overruns:** Traditional methods couldn't handle complex projects, leading to delays and exceeding budgets.
- **Buggy and Low-Quality Software that don't mee user requirements:** The existing practices resulted in software riddled with errors and often failing to meet user requirements.

  The complexity of codebases made them difficult to understand and modify, hindering future development.

The software crisis forced the field of computer science to re-evaluate its approach to software development. This led to the development of:

- **Software Engineering Principles:** A more disciplined approach to software development, emphasizing planning, design, and testing.
- **New Methodologies:** Agile development and DevOps emerged to address the need for flexibility and collaboration in software creation.

II. System development life cycle (SDLC)

SDLC is a framework that defines tasks performed at each step in the software development process.

It cans also be defined as a systematic process for building software that ensures the quality and correctness of the software built.

SDLC process aims to produce high-quality software that meets customer expectations.

1. System development life cycle (SDLC) Phases:

The SDLC has 7 standard phases which are:

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 requirements and recognition 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 within the legal framework of the area where the software will be used?
- ***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:

A. 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

B. 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 build 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 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

III. Software Development life cycle Models

An SDLC model is a particular approach of applying the SDLC on projects

1. **Waterfall Model:**
    - **Structure:** Sequential phases (Planning, Design, Development, Testing, Deployment, Maintenance) with strict progression. Each phase must be completed before moving to the next.
    - **Pros:** Simple to understand, good for well-defined projects with clear requirements.
    - **Cons:** Inflexible, difficult to adapt to changing needs, high risk of errors discovered late in the process.
2. **Agile Model:**
    - **Structure:** Iterative and incremental development. Focuses on delivering working software in short cycles with continuous feedback and improvement.

- **Pros:** Adaptable to changing requirements, faster time to market, better user feedback integration.
- **Cons:** Requires strong communication and collaboration, may not be suitable for projects with strict deadlines or complex requirements.

3. **Spiral Model:**
   - **Structure:** Combines elements of waterfall and iterative models. Focuses on risk management with iterative cycles of planning, risk identification, development, and evaluation.
   - **Pros:** Risk-driven approach, allows for early identification and mitigation of risks.
   - **Cons:** Can be complex to manage, requires experienced personnel for risk assessment.
4. **V-Model:**
   - **Structure:** Similar to waterfall but testing activities are planned in parallel with development phases (Verification for each Development stage & Validation at the end).
   - **Pros:** Strong focus on testing, ensures early identification of defects.
   - **Cons:** Less flexible than agile, may be overkill for smaller projects.
5. **Incremental Model:**
   - **Structure:** Delivers the software in small, working increments with each iteration building upon the previous one.
   - **Pros:** Manages complexity, allows for early user feedback and course correction.
   - **Cons:** Requires good planning and requirement breakdown, may lead to rework if requirements change significantly later.
6. **Prototype Model:**
   - **Structure:** Focuses on rapidly building a functional prototype to test and gather user feedback before full development.
   - **Pros:** Validates concepts and clarifies requirements early, reduces development risk.
   - **Cons:** Prototype may not reflect the final product, can be time-consuming to build a detailed prototype.
7. **Big Bang Model:**
   - **Structure:** Minimal planning, all functionalities are developed at once and then tested.
   - **Pros:** Quick development for very simple projects.
   - **Cons:** High risk of failure, difficult to manage complexity, not recommended for most projects.