SOFTWARE ENGINEERING

ASSIGNMENT

SUBMITTED BY :-
AKSHAT BATRA
2K18/IT/015

Assignment-1

Software Development Life Cycle Process

What is a Software Development Life Cycle Process (SDLC)?
A software life cycle model or process model is a pictorial and diagrammatic representation of the software's journey from inception to retirement. A life cycle model represents all the methods required to make a software product transit through its life cycle stages. It also captures the structure in which these methods are to be undertaken.

In other words, a life cycle model maps the various activities performed on a software product from its inception to retirement. Different life cycle models may plan the necessary development activities to phases in different ways. The essential activities are contained in all life cycle models though the action may be carried out in different orders in different life cycle models. During any life cycle stage, more than one activity may also be carried out.

Why is Software Development Life Cycle Process (SDLC) Needed?

- It offers a basis for project planning, scheduling, and estimating
- It provides a framework for a standard set of activities and deliverables
- It is a mechanism for project tracking and control
- Increases visibility of project planning to all involved stakeholders of the development process
- It increases and enhances development speed
- It improves client relations
- It helps to decrease project risk and project management plan overhead

Phases of SDLC

Various phases of Software Development Life Cycle (SDLC) have been detailed below:

Phase 1: **Requirement Collection & 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 as cyber law and other regulatory framework/compliances?
- Operation feasibility: Can we create operations which is expected by the client?
- Technical: Can the current computer system support the software?
- Schedule: Can the project be completed within the time constraints?

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:

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

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 (SDLC).

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

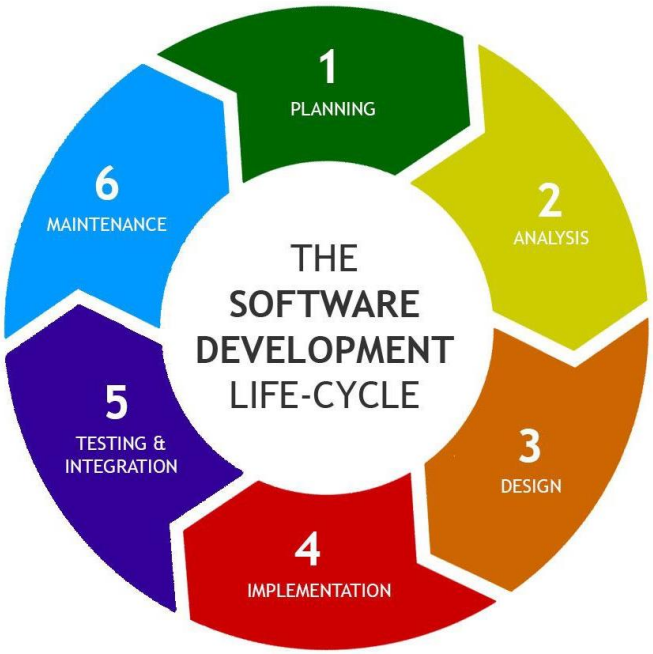
given by the project manager, the final software is released and checked for deployment issues if any.

Phase 7: **Maintenance**
Once the system is deployed, and customers start using the developed system, following three activities occur:

- Bug Fixes: Bugs are reported because of some scenarios that become visible only from mass use of software
- Upgrade: Upgrading the application to the newer versions of the software
- Enhancement: Adding some new features into the existing software

The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase.

Pictorial Representation of SDLC



Examples of SDLC Models
Some SDLC Models are:

1. Waterfall Model
2. Prototyping Model
3. Incremental Model
4. Iterative Enhancement Model
5. Spiral Model

Life Cycle Models: Their Advantages & Disadvantages

1. Waterfall Life Cycle Model

It is a sequential model that divides software development into pre-defined phases. Each phase must be completed before the next phase can begin with no overlap between the phases. Each phase is designed for performing specific activity during the SDLC phase. It was introduced in 1970 by Winston Royce.

Advantages:

- This model is simple to implement also the number of resources that are required for it is minimal
- The requirements are simple and explicitly declared; they remain unchanged during the entire project development
- The start and end points for each phase is fixed, which makes it easy to cover progress
- The release date for the complete product, as well as its final cost, can be determined before development

It gives easy to control and clarity for the customer due to a strict reporting system

Disadvantages:

- In this model, the risk factor is higher, so this model is not suitable for more significant and complex projects
- This model cannot accept the changes in requirements during development
- It becomes tough to go back to the phase. For example, if the application has now shifted to the coding phase, and there is a change in requirement,
  it becomes tough to go back and change it
- Since the testing done at a later stage, it does not allow identifying the challenges and risks in the earlier phase, so the risk reduction strategy is difficult to prepare

2. Prototyping Model

The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built. A prototype is a toy implementation of the system. A prototype usually turns out to be a very crude version of the actual system, possible exhibiting limited functional capabilities, low reliability, and inefficient performance as compared to actual software. In many instances, the client only has a general view of what is expected from the software product. In such a scenario where there is an absence of detailed information regarding the input to the system, the processing needs, and the output requirement, the prototyping model may be employed.

Advantages:

- Reduces the risk of incorrect user requirement
- Beneficial where requirements are changing/uncommitted
- Regular visible process aids management
- Supports early product marketing
- Reduces maintenance cost
- Errors can be detected much earlier as the system is made side by side

Disadvantages:

- An unstable/badly implemented prototype often becomes the final product
- Requires extensive customer collaboration
    - Costs customer money
    - Needs committed customer
    - Difficult to finish if customer withdraw
    - May be too customer specific, no broad market
- Difficult to know how long the project will last
- Easy to fall back into the code and fix without proper requirement analysis, design, customer evaluation and feedback
- Prototyping tools are expensive
- Special tools & techniques are required to build a prototype
- It is a time-consuming process

3. Incremental Model

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.

Advantages:

- Errors are easy to recognize
- Easier to test and debug
- More flexible
- Simple to manage risk because it handled during its iteration
- The client gets important functionality early

Disadvantages:

- Need for good planning
- Total Cost is high
- Well defined module interfaces are needed

4. Iterative Enhancement Model

In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

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 is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

Advantages:

- Testing and debugging during smaller iteration is easy
- A parallel development can be planned
- It is easily acceptable to ever-changing needs of the project
- Risks are identified and resolved during each iteration
- Limited time spent on documentation and extra time on designing

Disadvantages:

- Not suitable for smaller projects
- More resources may be required due to ever changing nature

- Design can be changed again and again because of imperfect requirements
- Project completion date cannot be confirmed

Spiral Model

The spiral model, initially proposed by Boehm, is an evolutionary software process model that couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model. It implements the potential for rapid development of new versions of the software. Using the spiral model, the software is developed in a series of incremental releases. During the early iterations, the additional release may be a paper model or prototype. During later iterations, more and more complete versions of the engineered system are produced.

Advantages:

- High amount of risk analysis
- Useful for large and mission-critical projects

Disadvantages:

- Can be a costly model to use
- Risk analysis needed with very particular expertise
- Doesn't work well for smaller projects