BOOKBIN: SOFTWARE DEVELOPMENT LIFE CYCLE MODEL

MD. NUROL AMIN

XYZ , Daffodil Smart City, Ashulia, Savar.

TABLE OF CONTENTS

1.0 Software Development Life Cycle (SDLC) 2
2.0 Software Development Methodologies 2
2.1 Waterfall Model 2
2.2 Spiral Model 3
2.3 Iterative Model 3
2.4 V-Model 3
2.5 Agile Mode 4
2.6 RAD (Rapid Application Development) Model 4
3.0 **Agile model** for BookBin application development 5
3.2 Principles and Requirements − 5
3.3 Agile Vs Traditional SDLC Models 6
3.4 The advantages of the Agile Model are as follows − 6
3.5 Agile vs Lean 7

1.0 SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

2.0 SOFTWARE DEVELOPMENT METHODOLOGIES

2.1 WATERFALL MODEL

The waterfall model uses a logical progression of SDLC steps for a project, similar to the direction water flows over the edge of a cliff. It sets distinct endpoints or goals for each phase of development. Those endpoints or goals can't be revisited after their completion.

The classical waterfall model divides the life cycle into a set of phases. This model considers that one phase can be started after the completion of the previous phase. That is the output of one phase will be the input to the next phase. Thus the development process can be considered as a sequential flow in the waterfall.

2.2 SPIRAL MODEL

The Spiral Model is a Software Development Life Cycle (SDLC) model that provides a systematic and iterative approach to software development. It is based on the idea of a spiral, with each iteration of the spiral representing a complete software development cycle, from requirements gathering and analysis to design, implementation, testing, and maintenance.

The Spiral Model provides support for Risk Handling. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a Phase of the software development process.

2.3 ITERATIVE MODEL

The Iterative model is all about repetition. Instead of starting out with a comprehensive overview of the requirements, development teams build the software piece by piece and identify further requirements as they go along. As a result, each new phase in the Iterative model produces a newer, more-refined version of the software under development.

Iterative allows developers and testers to identify functional or design flaws early, and can easily adapt to the ever-changing needs of the client. Like Spiral, Iterative suits larger projects and requires more management and oversight to work well.

2.4 V-MODEL

Similar to the waterfall methodology where testing is done at the end of a project, with the V-model, testing happens at every stage of development. The next stage of the V-model starts only when the previous stage is entirely finished.As part of the V-Model, a software tester has to verify if the requirements of a specific development phase are met. They also have to validate

that the system meets the needs of the user, customer or other stakeholders, which includes both verifications and validations.

2.5 AGILE MODE

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations

2.6 RAD (RAPID APPLICATION DEVELOPMENT) MODEL

Rapid application development is a software development methodology that uses minimal planning in favor of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product.

In the RAD model, the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery. Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process.

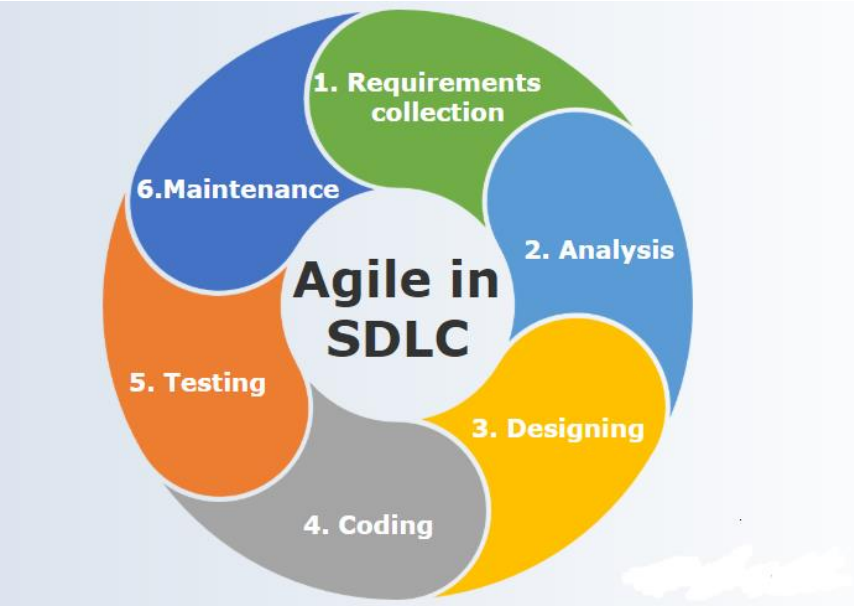
RAD projects follow iterative and incremental model and have small teams comprising of developers, domain experts, customer representatives and other IT resources working progressively on their component or prototype.

The most important aspect for this model to be successful is to make sure that the prototypes developed are reusable.

2.7 LEAN MODEL

Lean Software Development is a lightweight, Agile framework that focuses on reducing time and resources, eliminating waste, and ultimately delivering only what the product needs. The Lean approach is often applied as the Minimum Viable Product (MVP) technique, in which a team creates a barely functioning prototype and offers it to the market. The team then gathers feedback from customers about what they enjoy, dislike, and want to see improved and iterates based on this information.

3.0 **AGILE MODEL** FOR BOOKBIN APPLICATION DEVELOPMENT



Agile uses an adaptive approach where there is no detailed planning and there is clarity on future tasks only in respect of what features need to be developed. There is feature driven development and the team adapts to the changing product requirements dynamically. The product is tested very frequently, through the release iterations, minimizing the risk of any major failures in future.

Customer Interaction is the backbone of this Agile methodology, and open communication with minimum documentation are the typical features of Agile development environment. The agile teams work in close collaboration with each other and are most often located in the same geographical location

3.2 PRINCIPLES AND REQUIREMENTS −

- Individuals and interactions − In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.

- Working software − Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.
- Customer collaboration − As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
- Responding to change − Agile Development is focused on quick responses to change and continuous development.

3.3 AGILE VS TRADITIONAL SDLC MODELS

Agile is based on the adaptive software development methods, whereas the traditional SDLC models like the waterfall model is based on a predictive approach. Predictive teams in the traditional SDLC models usually work with detailed planning and have a complete forecast of the exact tasks and features to be delivered in the next few months or during the product life cycle.

Predictive methods entirely depend on the requirement analysis and planning done in the beginning of cycle. Any changes to be incorporated go through a strict change control management and prioritization.

Agile uses an adaptive approach where there is no detailed planning and there is clarity on future tasks only in respect of what features need to be developed. There is feature driven development and the team adapts to the changing product requirements dynamically. The product is tested very frequently, through the release iterations, minimizing the risk of any major failures in future.

Customer Interaction is the backbone of this Agile methodology, and open communication with minimum documentation are the typical features of Agile development environment. The agile teams work in close collaboration with each other and are most often located in the same geographical location.

3.4 THE ADVANTAGES OF THE AGILE MODEL ARE AS FOLLOWS −

- Is a very realistic approach to software development.
- Promotes teamwork and cross training.

- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required.
- Easy to manage.
- Gives flexibility to developers.

3.5 AGILE VS LEAN

- Frequent Change and Uncertainty: Agile is particularly effective in environments where requirements change frequently, and there is a need for flexibility and rapid adaptation to customer feedback.

- Customer Collaboration: Having direct access to customers or end-users who can provide continuous feedback and you want to involve them closely in the development process, Agile's customer-centric approach is beneficial.

- Iterative and Incremental Delivery: Delivering incremental value in short iterations aligns with your project's goals and stakeholder expectations, Agile is a good choice.

- Cross Functional Team : Agile teams are typically cross-functional, with members responsible for various aspects of the app.

- Adaptability : Well suited to projects with evolving requirements. Allows for flexibility in responding to changing user needs.

- Provides a dynamic environment with evolving user needs .

4.0 CONTACT

By Phone : +8801721716297

By Email: xyz@diu.edu.bd

Website: www.xyz.com